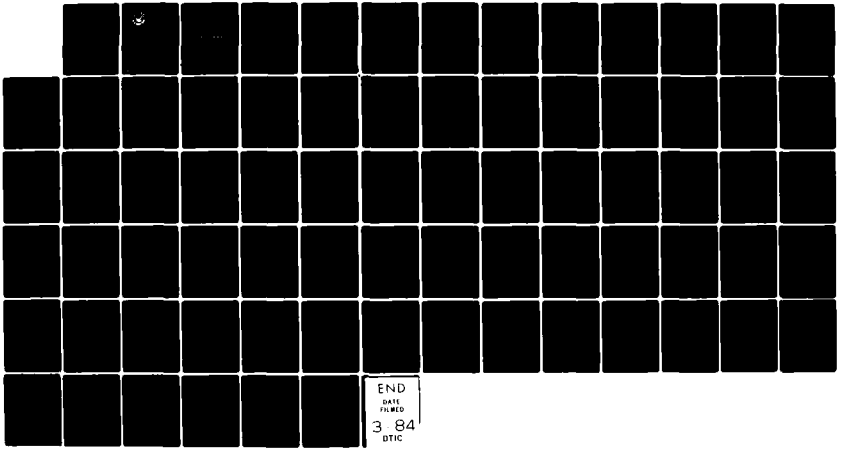
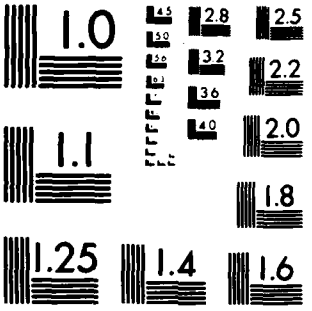
AD-A137 646 COMPUTER IMPLEMENTATION OF THE BOUNDING SURFACE PLASTICITY MODEL FOR COHESIVE SOILS(U) CALIFORNIA UNIV DAVIS L R HERRMANN ET AL. DEC 83 NCEL-CR-84.007 1/1

UNCLASSIFIED N62583-83-M-T062 F/G 8/13 NL



END
DATE FILMED
3 84
DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AD A137646

(12)

CR 84.007

NAVAL CIVIL ENGINEERING LABORATORY
Port Hueneme, California

Sponsored by
NAVAL FACILITIES ENGINEERING COMMAND

COMPUTER IMPLEMENTATION OF THE BOUNDING SURFACE PLASTICITY MODEL FOR COHESIVE SOILS

December 1983

An Investigation Conducted by
UNIVERSITY OF CALIFORNIA, DAVIS

DTIC
ELECTE
FEB 9 1984
A

N62583-83-M-T062

DTIC FILE COPY

Approved for public release; distribution unlimited.

84 02 09 048

METRIC CONVERSION FACTORS

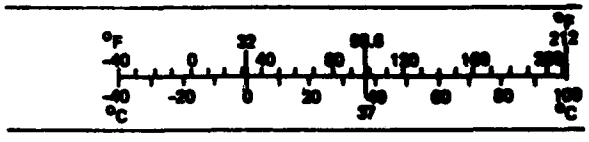
Approximate Conversions to Metric Measures

Symbol	When You Know	Multiply by	To Find	Symbol
		LENGTH		
in	inches	*2.5	centimeters	cm
ft	feet	30	centimeters	cm
yd	yards	0.9	meters	m
mi	miles	1.6	kilometers	km
		AREA		
in^2	square inches	6.5	square centimeters	cm^2
ft^2	square feet	0.09	square meters	m^2
yd^2	square yards	0.8	square meters	m^2
mi^2	square miles	2.6	square kilometers	km^2
	acres	0.4	hectares	ha
		MASS (weight)		
oz	ounces	28	grams	g
lb	pounds	0.45	kilograms	kg
	short tons (2,000 lb)	0.9	tonnes	t
		VOLUME		
tsp	teaspoons	5	milliliters	ml
Tbsp	tablespoons	15	milliliters	ml
fl oz	fluid ounces	30	milliliters	ml
c	cups	0.24	liters	l
pt	pints	0.47	liters	l
qt	quarts	0.95	liters	l
gal	gallons	3.8	liters	l
ft^3	cubic feet	0.03	cubic meters	m^3
yd^3	cubic yards	0.76	cubic meters	m^3
		TEMPERATURE (exact)		
°F	Fahrenheit temperature	5/9 (after subtracting 32)	Celsius temperature	°C

*1 in = 2.54 (exactly). For other exact conversions and more detailed tables, see NBS Misc. Publ. 286, Units of Weights and Measures, Price $2.25, SD Catalog No. C13.10:286.

Approximate Conversions from Metric Measures

Symbol	When You Know	Multiply by	To Find	Symbol
		LENGTH		
mm	millimeters	0.04	inches	in
cm	centimeters	0.4	inches	in
m	meters	3.3	feet	ft
m	meters	1.1	yards	yd
km	kilometers	0.6	miles	mi
		AREA		
cm^2	square centimeters	0.16	square inches	in^2
m^2	square meters	1.2	square yards	yd^2
km^2	square kilometers	0.4	square miles	mi^2
ha	hectares (10,000 m^2)	2.5	acres	
		MASS (weight)		
g	grams	0.035	ounces	oz
kg	kilograms	2.2	pounds	lb
t	tonnes (1,000 kg)	1.1	short tons	
		VOLUME		
ml	milliliters	0.03	fluid ounces	fl oz
l	liters	2.1	pints	pt
l	liters	1.06	quarts	qt
l	liters	0.26	gallons	gal
m^3	cubic meters	35	cubic feet	ft^3
m^3	cubic meters	1.3	cubic yards	yd^3
		TEMPERATURE (exact)		
°C	Celsius temperature	9/5 (then add 32)	Fahrenheit temperature	°F



Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1 REPORT NUMBER CR 84.007	2. GOVT ACCESSION NO. AD-A137 646	3 RECIPIENT'S CATALOG NUMBER
4 TITLE (and Subtitle) Computer Implementation of the Bounding Surface Plasticity Model for Cohesive Soils		5 TYPE OF REPORT & PERIOD COVERED Final Jan 1983 - Oct 1983 6 PERFORMING ORG. REPORT NUMBER
7 AUTHOR(s) L.R. Herrmann, V.N. Kaliakia, V.F. Dafalias		8 CONTRACT OR GRANT NUMBER(s) N62583-83-M-T062
9 PERFORMING ORGANIZATION NAME AND ADDRESS University of California, Davis		10 PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS YF023.03.01.002
11 CONTROLLING OFFICE NAME AND ADDRESS Naval Civil Engineering Laboratory Port Hueneme, CA 93043		12 REPORT DATE December 1983 13 NUMBER OF PAGES
14 MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Naval Facilities Engineering Command 200 Stovall Street Alexandria, VA 22332		15 SECURITY CLASS. (of this report) Unclassified 15a DECLASSIFICATION/DOWNGRADING SCHEDULE

16 DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17 DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18 SUPPLEMENTARY NOTES

19 KEY WORDS (Continue on reverse side if necessary and identify by block number)

Finite element, computer program, geotechnical engineering, soil constitutive law

20 ABSTRACT (Continue on reverse side if necessary and identify by block number)

The equations governing the consolidation, and the stress and strains states for soil structures are reviewed and their historical development is discussed. Numerical analysis concepts are used to express these equations in incremental form. A variational statement of these incremental equations is formulated and used in the development of a comprehensive

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

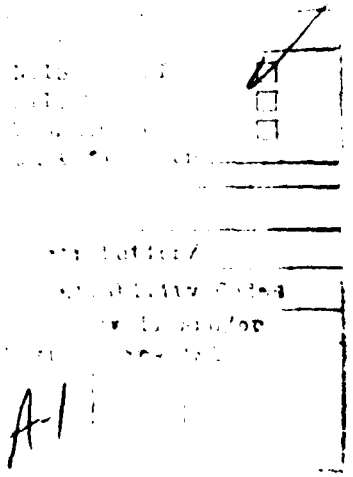
III

finite element analysis. The concepts used in developing the variational statement are somewhat different from those used by most other investigators and appear to offer certain advantages for inelastic formulations. Finally results obtained with the finite element analysis are compared to known solutions with good results.

For the convenience of the reader the total report on the project is presented in four parts. As noted above a description of the consolidation theory and certain theoretical features of the finite element analysis are described in the body of the main report (CR 84.006). The second part (CR 84.007) describes the numerical evaluation of the incremental form of the bounding surface model. Finally "user's manuals" for the 2-D and 3-D finite element programs are given in two additional reports (CR 84.008 and CR 84.009).

IV

TABLE OF CONTENTS



INTRODUCTION

During the past four years, extensive theoretical, experimental and numerical work has been done at UCD on a bounding surface plasticity model for cohesive soils. The work has been conducted in part under the sponsorship of contracts and grants from the U.S. Army Waterways Experiment Station (Re: Contract #DACA 39-79-M-0059), the Civil Engineering Laboratory, Naval Construction Battalion Center (Re: Orders N62583-80-M-R478, N62583-81-M-R320, N62474-82-C-8276 and N62583-83-M-T062), and NSF (Re: Grants NSF-CME-79-10835 and NSF-CEE-82-16995). To date this work has resulted in the reports and papers listed in references [1 → 14].

Although continued research will most certainly bring refinements and new areas of application, the model has now reached a stage of development where it can be used as a practical engineering tool. Such use requires that it be incorporated into new and existing finite element codes for the analysis of earth structures. In order to simply and inexpensively achieve such numerical implementations, the model has been coded into a master subroutine CLAY and several supporting subroutines [2,5].

Because of the evolutionary nature of the development of this code, with time it has become increasingly inefficient and unwidely. Thus, to improve its readability, to incorporate some recent minor revisions in the model and its numerical implementation, and to take advantage of the structured nature of FORTRAN 77, the subroutines have been recoded. In this process it was found to be convenient to slightly modify the calling instructions for the subroutine and thus there is a need to revise the instructions for its implementation as given in [5].

It is the purpose of this report to discuss the recent minor changes in the model, and its numerical implementation, and to give detailed instructions

for its incorporation into new and existing finite element codes. In addition, the program (EVAL) [10] written to evaluate homogeneous test results for cohesive soils is discussed. This report is for the most part a replacement for Section 2 and Appendices I and II of [5]. Frequent reference will be made to equations and figures from [5]; such references are expressed in the form (5-N), etc.

1. Recent Modifications to the Bounding Surface Plasticity Model for Cohesive Soils

This report is not intended to stand alone, but rather to be a replacement for certain sections of [5]. Thus, for a comprehensive description of the underlying theory, notation and the background for this report, the reader is referred to [5].

In this section the modifications that have been made to the bounding surface plasticity model for cohesive soils since the publication of [5] are discussed.

Modification 1:

Following a recent paper by Dafalias [15], the "radial" mapping rule of eq. (5-12) has been generalized to give an image stress state of

$$\bar{I} = \beta(I - I_c) + I_c, \quad \bar{s}_{ij} = \beta s_{ij}, \quad \bar{J} = \beta J, \quad \bar{\alpha} = \alpha \qquad (1)$$

This generalization has introduced the possibility of using a projection center ($I_c = CI_o$) in stress space different than the origin. This is illustrated in Fig. 1; note that the parameter "C" has no relation to point "C" of Figure 5-2. For C = 0 one retrieves the previous formulation. This modification introduces a kind of "hydrostatic back stress," I_c, and allows for the prediction of immediate negative pore pressure development for heavily overconsolidated samples under undrained loading conditions. With the projection center at the stress origin,

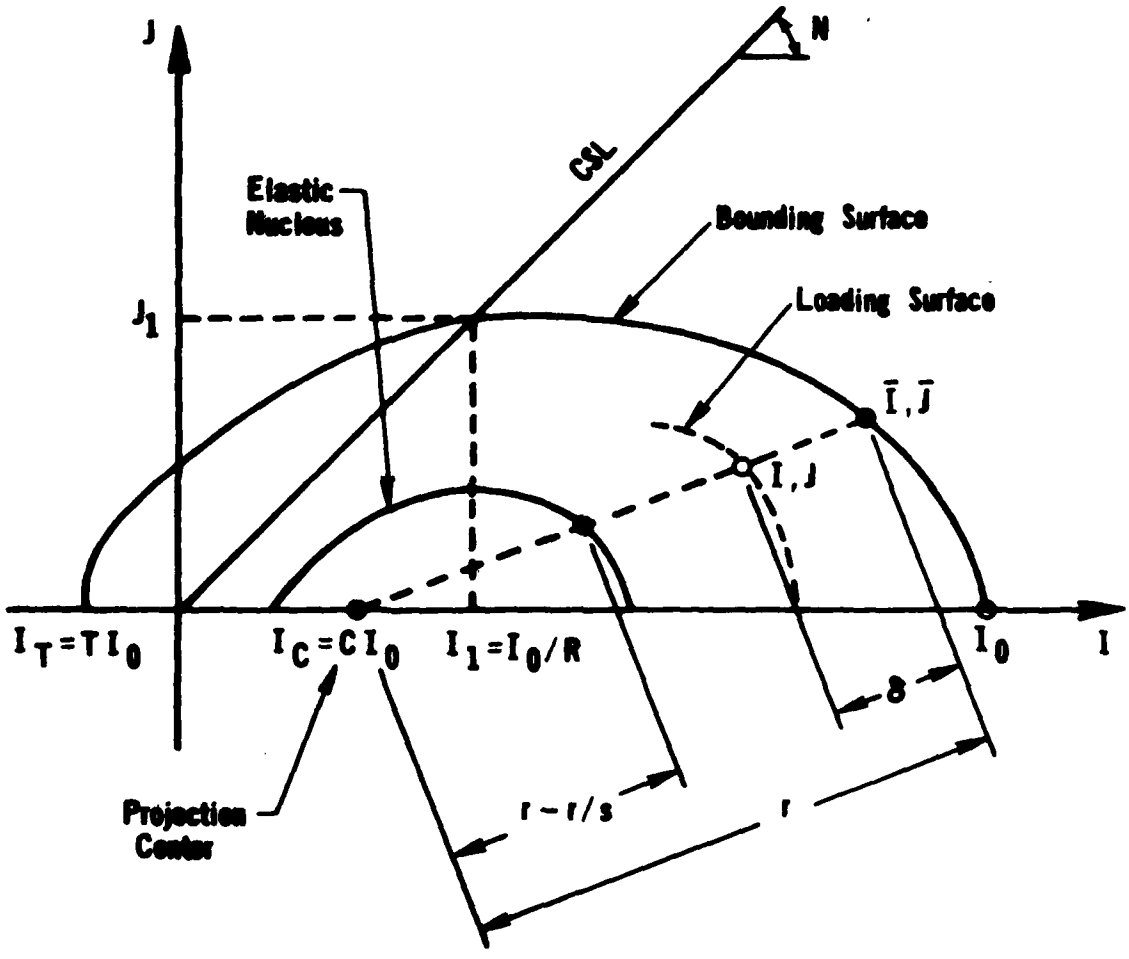


Figure 1. Schematic Illustration of the Radial Mapping Model in Invariant Stress Space Showing the Projection Center and the Elastic Nucleus.

one would always have an initial positive porewater pressure built-up even for large OCR values.

Modification 2:

The denominator of $\bar{r} - \delta$ in eq. (5-53) has been replaced by the quantity $\langle r - s\delta \rangle$ with s being an elastic nucleus parameter and the symbol $\bar{r}$ being now simply written as r. Thus, whenever $\delta > r/s$ the brackets yield a zero value and according to eq. (5-28) $K_p = \infty$. Observe that $K_p \to \infty$ as $\delta \to r/s$. A geometric interpretation of this behavior is that s indirectly defines a stress domain of purely elastic response within the bounding surface; it is not necessary to interpret the boundary of this zone as a yield surface (no associated consistancy condition, etc.). This domain is called the "elastic nucleus" and is shown in Fig. 1. The quantity s which controls the size of the elastic nucleus can be a function of the state including the accumulated deviatoric plastic strains. The elastic nucleus controls the possible stabilization of cyclic undrained stress paths, allowing for stabilization or failure depending upon the amplitude of the cyclic deviatoric stress.

Modification 3:

Eq. (5-55) is replaced by

$$\frac{dI_o}{de^n} = - \frac{\langle I_o - I_\ell \rangle + I_\ell}{\lambda - \kappa} \qquad (2)$$

This expression predicts that for tensile stresses the soil looses all cohesion at a maximum but finite void ratio. The original expression, eq. (5-55), implied a plastic void ratio tending towards infinity as $I_o \to 0$. (Recall that I_o is the internal variable controlling the size of the bounding surface and, in general, is not the current value of I).

Modification 4:

The quantity p_a (atmospheric pressure) in eq. (5-84) has been replaced by $(1 + e_o)[\langle I_o - I_\ell \rangle + I_\ell]/(\lambda - \kappa)$. The term I_o is now used (instead of p_a) to

non-dimensionalize $h(\alpha)$ and the term $(1 + e_o)/(\lambda - \kappa)$ is introduced for reasons of convenience since $\bar{K}_p$ depends on this factor as well. The use of I_o in this expression renders all undrained stress paths at a given OCR similar when normalized with respect to I_o. If natural strains are used instead of engineering strains, the inital void ratio (e_o) is replaced by the current value (e).

Modification 5:

The term $h(\alpha)$ in eq. (5-84) has been replaced by

$$h(\alpha,z) = z^m h_1(\alpha) + (1 - z^m) h_2$$

where (see Fig. 1 and refer to eq. (5-58))

$$h_1(\alpha) = \frac{2\mu}{1 + \mu - (1-\mu) \sin 3\alpha} h_c$$

$\mu = \frac{h_e}{h_c}$, with h_e, h_c being the values of h for triaxial compression and extension respectively.

$$z = \frac{J}{J_1} = \frac{J}{\sqrt{3} I_1} = \frac{\sqrt{3} R}{\sqrt{3} I_o}$$

m = a constant which applies to both extension and compression and is thus not a function of the Lode angle α.

h_2 = the shape hardening parameter for states on the I-axis (i.e. for $z = 0$).

This term assures continuity when crossing the I-axis; the modification was incorporated to improve numerical behavior in this region, however, it has very little affect on the model predictions.

Incorporating modifications 2, 4 and 5, eq. (5-84) becomes

$$K_p = \bar{K}_p + \frac{(1 + e_o)}{\lambda - \kappa} [\langle I_o - I_\ell \rangle + I_\ell][z^m h_1 + (1 - z^m)h_2]$$
$$[9 F_{,\bar{I}} + \tfrac{1}{3} F_{,\bar{J}}][\frac{\delta}{\langle r - s\delta \rangle}] \qquad (3)$$

The model as described in [5] and subject to the above modifications gives a comprehensive description of the three-dimensional stress-strain behavior of cohesive soils. The practical use of the model depends upon its numerical implementation in new and existing finite element codes for earth structures; this topic will be discussed in the following section.

2. NUMERICAL IMPLEMENTATION*

2.1 Incrementalization of the Bounding Surface Plasticity Rate Equations

The bounding surface plasticity theory for cohesive soils is expressed in terms of effective stress, whereas most soil related problems involve the application and calculation of total stress. The difference between the total and effective stress is simply the pore water pressure u. This section deals with the incremental relation between the strain and effective stress components; the pore water pressure, and hence the total stress, is dealt with in a following section.

Factoring the strain increment from eq. (5-30) gives:

$$\dot{\sigma}_{ij} = D_{ijk\ell}\, \dot{\epsilon}_{k\ell} \tag{4}$$

where

$$D_{ijk\ell} = G(\delta_{ki}\delta_{\ell j} + \delta_{kj}\,\delta_{\ell i}) + (K - \tfrac{2}{3}G)\delta_{ij}\,\delta_{k\ell} - \Bigg[3K\,F,_{\bar{I}}\,\delta_{ij} + \frac{G}{\bar{J}}F,_{\bar{J}}\,s_{ij} + \frac{\sqrt{3}\,G}{\bar{J}\cos 3\alpha}F,_{\alpha}\left(\frac{s_{in}\,s_{nj}}{J^2} - \frac{3\,S^3 s_{ij}}{2\,J^4} - \frac{2\,\delta_{ij}}{3}\right)\Bigg]$$
$$\Bigg[3K\,F,_{\bar{I}}\,\delta_{k\ell} + \frac{G}{\bar{J}}F,_{\bar{J}}\,s_{k\ell} + \frac{\sqrt{3}\,G}{\bar{J}\cos 3\alpha}F,_{\alpha}\left(\frac{s_{kn}\,s_{n\ell}}{J^2} - \frac{3\,S^3 s_{k\ell}}{J^4} - \frac{2\,\delta_{k\ell}}{3}\right)\Bigg]\mathcal{L}_{/n} \tag{5}$$

---

* This section serves as a replacement for the corresponding section in [5].

where

$$\mathcal{L} = \begin{cases} 1 & L > 0 \quad \text{(loading)} \\ 0 & L \leq 0 \quad \text{(unloading)} \end{cases} \tag{6}$$

$$L = \frac{1}{D}\left\{3K F_{,\bar{I}}\,\dot{\epsilon}_{kk} + \frac{G}{\bar{J}} F_{,\bar{J}}\, s_{ij}\,\dot{\epsilon}_{ij} + \frac{\sqrt{3}\,G}{\bar{J}\cos 3\alpha} F_{,\alpha}\left[\left(\frac{s_{ik}\,s_{kj}}{J^2} - \frac{3\,S^3 s_{ij}}{2\,J^4}\right)\dot{\epsilon}_{ij} - \frac{2\,\dot{\epsilon}_{kk}}{3}\right]\right\} \tag{7}$$

$$D = K_p + 9K\left(F_{,\bar{I}}\right)^2 + G\left(F_{,\bar{J}}\right)^2 + \frac{G}{\bar{J}^2}\left(F_{,\alpha}\right)^2 \tag{8}$$

Eq. (4) relates the tensor components of (effective) stress and strain. For finite element analysis purposes, it is more convenient to express this relationship in matrix form, i.e.,

$$\{\dot{\sigma}\} = [D]\ \{\dot{\epsilon}\} \tag{9}$$

where $([\ ]^T$ is the matrix transpose)

$$\{\sigma\}^T = (\sigma_x,\ \sigma_y,\ \sigma_z,\ \tau_{xy},\ \tau_{xz},\ \tau_{yz}) \tag{10}$$
$$\{\epsilon\}^T = (\epsilon_x,\ \epsilon_y,\ \epsilon_z,\ \gamma_{xy},\ \gamma_{xz},\ \gamma_{yz})$$

The tensor components of shear strain ϵ_{ij} are one-half of the engineering components γ_{ij}. The [D] matrix is related to the components of the $D_{ijk\ell}$ tensor as follows (because of the symmetry of the stress and strain tensors, interchanging i and j or k and ℓ in eq. (5) results in the same quantity):

$$[D] = \begin{bmatrix} D_{1111} & D_{1122} & D_{1133} & D_{1112} & D_{1113} & D_{1123} \\ & D_{2222} & D_{2233} & D_{2212} & D_{2213} & D_{2223} \\ & & D_{3333} & D_{3312} & D_{3313} & D_{3323} \\ & & & D_{1212} & D_{1213} & D_{1223} \\ & (\text{symm}) & & & D_{1313} & D_{1323} \\ & & & & & D_{2323} \end{bmatrix}$$

In order to use eq. (9) in a finite element program, it must be expressed in an incremental form. Consider the Nth step of an incremental analysis; i.e., the solution has been found at N-1, and it is now desired to calculate the incremental change that will give the solution at N.

Numerous numerical methods have been developed for calculating incremental solutions to plasticity problems. A simple family of solutions which can be viewed as approximations to Newton-Raphson's method will be discussed here [12,16]; members of this family include the classical method of "successive approximations" and the popular "tangent stiffness method" [16]. With little additional effort the equations to be discussed can be adapted for use with a number of other methods.

Because of the nonlinear nature of elastic-plastic behavior, iteration is in general required to establish the incremental change. At the end of the K-1 iteration, the estimates of the stress and strain states at N are given by the expressions:

$$\{\sigma\}_{N,K-1} = \{\sigma\}_{N-1} + \{\Delta\sigma\}_{N,K-1} \tag{11}$$

$$\{\varepsilon\}_{N,K-1} = \{\varepsilon\}_{N-1} + \{\Delta\varepsilon\}_{N,K-1} \tag{12}$$

The iteration process is continued until some specified convergence criterion is satisfied, or until a specified maximum number of iterations have

failed to yield convergence (an indication of possible failure and/or unstable behavior).

Even though rate independent behavior is being considered, it is convenient to think in terms of the time history of the quantities involved. If eq. (9) is integrated from time t_{N-1} to t_N, it yields:

$$\int_{t_{N-1}}^{t_N} \{\dot{\sigma}\}\, dt = \int_{t_{N-1}}^{t_N} [D]\, \{\dot{\epsilon}\}\, dt \tag{13}$$

or

$$\{\Delta\sigma\}_N = \int_{t_{N-1}}^{t_N} [D]\, \{\dot{\epsilon}\}\, dt \tag{14}$$

A simple two point formula is used to approximate the above integral, $(0 \leq \theta_1 \leq 1)$:

$$\{\Delta\sigma\}_N = \left[(1 - \theta_1)\ [D]_{N-1} + \theta_1\ [D]_N\right] \{\Delta\epsilon\}_N \tag{15}$$

Values of $\theta_1 = 0$, 1/2 and 1 give, respectively, forward, trapezoidal and backward integration. Although trapezoidal integration is most accurate, in rare instance it may be advantageous to use $\theta_1 = 0.0$ in order to reduce iteration requirements (at the expense of smaller step sizes).

Because $[D]_N$ and $[D]_{N-1}$ are functions of the stress and strain states at N (see eqs. (5,6)), it is necessary to base their values on the stress and strain estimates of the previous iteration (see eqs. (11) and (12)). The fact that $[D]_{N-1}$ is possibly a function of $\{\Delta\sigma\}_N$ and $\{\Delta\epsilon\}_N$ requires some explanation. The dependence is a result of the discontinuous nature of plasticity behavior at the initiation of an unloading process (see eq. (6)). For the simple one-dimensional example shown in Figure 2, the stiffness D_{N-1} will be D' or D" depending upon whether the step is to N' or N". While in a one-dimensional problem this would

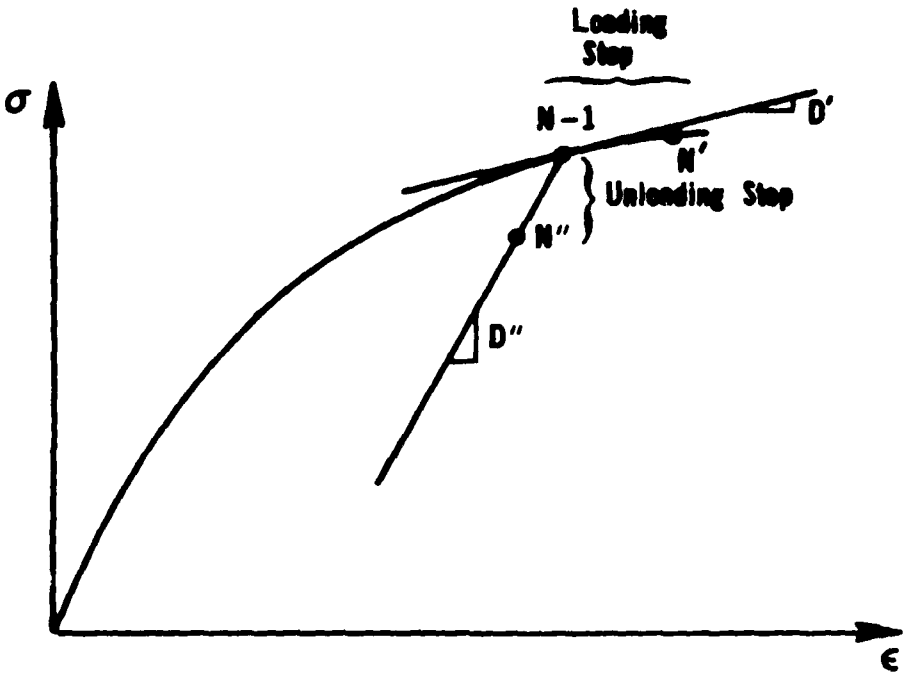


Figure 2. Illustration of the Discontinuity in Stiffness at the Initiation of Unloading.

usually be known a-priori, for a finite element that is a part of a complicated, highly statically indeterminate, two or three-dimensional structure it can only be established by the iteration process. If this dependence of $[D]_{N-1}$ on the iteration process is ignored and θ_1 is taken as zero, then iteration is not necessary, however, very small time steps are required for accuracy; in general, this procedure is not recommended. The predicted value for $[D]_N$ is denoted by $[D]_{N,K-1}$ etc. The equation resulting from substituting these estimates into eq. (15) is used to relate the estimates of $\{\Delta\sigma\}_N$ and $\{\Delta\varepsilon\}_N$ for iteration K, i.e.,

$$\{\Delta\sigma\}_{N,K} = [\bar{D}]_{N,K-1} \{\Delta\varepsilon\}_{N,K} \tag{16}$$

where

$$[\bar{D}]_{N,K-1} = [(1 - \theta_1)[D]_{N-1,K-1} + \theta_1 [D]_{N,K-1}] \tag{17}$$

Eq. (16) is the desired incremental stress-strain relation for iteration K of increment N, and in the Newton-Raphson method, is used for the calculation of the residual vector [12]. In addition the Newton-Raphson method requires the Jacobian matrix. An approximation to the Jacobian can be written in the form [12,14].

$$[J]_{N,K-1} \simeq [(1 - \theta_2)[D]_{N-1,K-1} + \theta_2[D]_{N,K-1}] \tag{18}$$

Values of θ_2 of 1/2 and 1 correspond respectively to the methods of successive approximation and tangent stiffness [12,16]. The simple test evaluation program EVAL discussed in [10] and a later section of this report uses $\theta_2 = 1/2$, the consolidation codes discussed in [14] permit θ_2 to be specified by the user; all these applications use $\theta_1 = 1/2$ in eq. (17).

Thus, for finite element implementation what is required is the calculation of the matrices $[D]_{N-1,K-1}$ and $[D]_{N,K-1}$. The combining of these matrices in eq. (17) and (18) (or their use in some other fashion for a different nonlinear

solution algorithm) is left for the finite element program which calls the master subroutine CLAY that has been written for their evaluation. The main goal of this report is a description of the steps necessary for incorporating this subroutine into new or existing finite element codes (Section 2.3). The calculation of $[D]_{N-1,K-1}$ and $[D]_{N,K-1}$ proceeds directly from eq. (5) with only a few steps needing elaboration.

The parameter θ in eq. (5-50) is undefined for a zero value of the first effective stress invariant I. The numerical problems associated with a zero or near zero value of I are avoided by arbitrarily replacing the value of $|I|$ by $10^{-4}P_a$ for such cases (where P_a is atmospheric pressure); the corresponding expression in the equation of "Modification 5" (Section 1) is treated in a similar fashion. In general, this arbitrary action does not significantly affect the calculated properties.

As the soil state approaches the bounding surface, a stress state outside of the surface may be predicted in a particular iteration. Because such a prediction has no meaning, the state is assumed instead to fall on the surface; i.e., β is restricted to be ≥ 1.0 and $\frac{\delta}{<r-s\delta>}$ is restricted to be ≥ 0.

At any instant, the size of the bounding surface is determined by the value of I_o (see Figure 1). The differential change in I_o is obtained from eq. (2), i.e.,

$$dI_o = \frac{1+e_o}{\lambda-\kappa} (<I_o - I_\ell> + I_\ell)(d\epsilon_{kk} - \frac{1}{3K} dI) \qquad (19)$$

Two cases must now be considered in integrating (19): If $I_o > I_\ell$, eq. (19) becomes:

$$dI_o = (1+e_o) \frac{I_o}{\lambda-\kappa} (d\epsilon_{kk} - \frac{1}{3K} dI) \qquad (20)$$

Dividing by I_o and integrating the resulting expression for increment N gives:

$$\int_{I_{o_{N-1}}}^{I_{o_{N,K-1}}} \frac{dI_o}{I_o} = \frac{(1+e_o)}{\lambda-\kappa} \left[\int_{\varepsilon_{kk_{N-1}}}^{\varepsilon_{kk_{N,K-1}}} d\varepsilon_{kk} - \int_{I_{N-1}}^{I_{N,K-1}} \frac{1}{3K} \right] dI \qquad (21)$$

The first two integrals may be evaluated exactly, while the third is approximated by the trapezoidal rule giving:

$$\ell n \left(\frac{I_{o_{N,K-1}}}{I_{o_{N-1}}} \right) = \frac{(1+e_o)}{\lambda-\kappa} \left[\Delta\varepsilon_{kk_{N,K-1}} - \frac{1}{6} \left(\frac{1}{K_{N-1}} + \frac{1}{K_{N,K-1}} \right) \Delta I_{N,K-1} \right] \qquad (22)$$

or

$$I_{o_{N,K-1}} = I_{o_{N-1}} \exp \left\{ \frac{(1+e_o)}{\lambda-\kappa} \left[\Delta\varepsilon_{kk_{N,K-1}} - \frac{1}{6} \left(\frac{1}{K_{N-1}} + \frac{1}{K_{N,K-1}} \right) \Delta I_{N,K-1} \right] \right\} \qquad (23)$$

If $I_o \leq I_\ell$, eq. (19) becomes

$$dI_o = (1+e_o) \frac{I_\ell}{\lambda-\kappa} \left(d\varepsilon_{kk} - \frac{1}{3K} dI \right) \qquad (24)$$

integrating the resulting expression for increment N gives:

$$\int_{I_{o_{N-1}}}^{I_{o_{N,K-1}}} dI_o = \frac{(1+e_o)}{\lambda-\kappa} I_\ell \left\{ \int_{\varepsilon_{kk_{N-1}}}^{\varepsilon_{kk_{N,K-1}}} d\varepsilon_{kk} - \int_{I_{N-1}}^{I_{N,K-1}} \frac{1}{3K} dI \right\} \qquad (25)$$

Evaluating the first two integrals exactly and approximating the third by the trapezoidal rule gives:

$$I_{o_{N,K-1}} = I_{o_{N-1}} + \frac{(1+e_o)}{\lambda-\kappa} I_\ell \left[\Delta\varepsilon_{kk_{N,K-1}} - \frac{1}{6} \left(\frac{1}{K_{N-1}} + \frac{1}{K_{N,K-1}} \right) \Delta I_{N,K-1} \right] \qquad (26)$$

Because the point of switching from eq. (20) to eq. (24) (controlled by the value of I_ℓ) is somewhat arbitrary, consideration was not given to the possibility of

this changeover occurring in mid increment. Instead the value of $I_{o_{N-1}}$ is used to make the decision for the whole increment.

Until convergence occurs, the estimates of $\{\Delta\sigma\}_{N,K-1}$ and $\{\Delta\epsilon\}_{N,K-1}$ used in the calculation of the incremental properties $[\bar{D}]_{N,K-1}$ do not in fact satisfy the incremental stress-strain relation, eq. (16). Because this inconsistency disappears as global convergence occurs (i.e., as $[\bar{D}]_{N,K-1} \simeq [\bar{D}]_{N,K-2}$), it is not absolutely necessary to take any special steps to avoid it, however, numerical experimentation has indicated computational advantages in doing so [12]. Thus, local (i.e., within subroutine CLAY each time it is called) iteration is introduced in the calculation of $[\bar{D}]$ to remove the inconsistency [17]. Using $\{\Delta\epsilon\}_{N,K-1}$ and $\{\Delta\sigma\}_{N,K-1}$, $[\bar{D}]_{N,K-1}$ is calculated. The values of $\{\Delta\epsilon\}_{N,K-1}$ and $[\bar{D}]_{N,K-1}$ are then used in conjunction with eq. (16) to calculate a new estimate of stress $\{\Delta\sigma\}^{*}_{N,K-1}$ which is in turn used with $\{\Delta\epsilon\}_{N,K-1}$ to calculate a new estimate for the incremental properties $[\bar{D}]^{*}_{N,K-1}$. This process is continued until convergence is achieved for $\{\Delta\sigma\}^{*}_{N,K-1}$. Because of the global iteration (the basic iterative procedure of the calling finite element program) which also tends to remove the inconsistency, it appears that the convergence limit on the local iteration can be considerably less restrictive than the global requirement.† The stress estimate is iteratively modified (instead of the strain estimate) in order to maintain a compatible global displacement field as required by the admissibility conditions of the finite element procedure. The introduction of local iteration (for all points in the finite element grid where the incremental stress-strain properties are required) of course substantially increases, in a given global iteration, the computational time for subroutine CLAY, however, there is a corresponding reduction in the number of global iterations.

---

† In [14], ten times the global limit is used.

2.2 Calculation of Pore Water Pressure

Many finite element programs have special provisions for handling pore water pressure calculations [14]. For such situations, the soil properties subroutine need only supply a relation between the increments of strain and effective stress, i.e., eqs. (16) and (18). For such cases the remainder of this section has no significance.

There are three possibilities concerning the development of pore water pressure in soil: ideal drained conditions (where the pore water pressure is identically zero), ideal undrained conditions (where the soil is completely saturated, and no flow of water occurs whatsoever), and the more realistic situation where there is a global flow of water and/or the filling of voids. In many analyses ideal drained or undrained conditions are assumed, even though they may only be approximately true.

The total stress rate $\dot{\sigma}^t_{ij}$ is the sum of the effective stress rate and the pore water pressure rate:

$$\dot{\sigma}^t_{ij} = \dot{\sigma}_{ij} + \dot{u}\ \delta_{ij} \tag{27}$$

For drained conditions u=0 and $\sigma^t_{ij} = \sigma_{ij}$, and eqs. (16) and (18) are the desired relations between the total stress increment and the strain increment.

For undrained conditions there are several possible ways of proceeding. The traditional approach has been to neglect the (slight) compressibility of the water and the soil particles, and thus assume incompressible material behavior. However, the finite element analysis of incompressible materials requires a special formulation [18,19,20].

In order to avoid having to deal with separate formulations for drained and undrained conditions, it is often convenient to express them in a common form (the numerical consequences of this step are discussed below). This can

be accomplished if the slight compressibility of the soil particles and the pore water is recognized [21]*. Thus, the pore water pressure u is written in terms of the combined bulk modulus Γ of the soil particles and the pore water and the resulting (very small) volume change ϵ_{kk} (note that as $\Gamma \to \infty$ the soil becomes incompressible, and that drained conditions are obtained when $\Gamma = 0$):

$$u = \Gamma \, \epsilon_{kk} \tag{28}$$

For undrained conditions the value of Γ is very large compared to the terms in $[\bar{D}]_{N,K-1}$. Thus, the soil behaves as a "nearly incompressible solid" [18,19], and, consequently, care must be exercised to avoid numerical round-off and element "locking" problems. Two approaches are commonly used to achieve this goal. One method is to use the special formulation given in references [18,19] for incompressible and nearly incompressible solids, while the other is to use "reduced" or "selective-reduced" integration [22,23] for the element stiffness matrix (the importance of selecting a proper element type is discussed in [20,22,24]). In the latter case, eq. (28) is used to eliminate $\dot{u}$ from eq. (27); i.e.,

$$\dot{\sigma}^t_{ij} = \dot{\sigma}_{ij} + \Gamma \, \dot{\epsilon}_{kk} \delta_{ij} \tag{29}$$

Integration over increment N gives:

$$\Delta\sigma^t_{ij_N} = \Delta\sigma_{ij_N} + \Gamma \, \Delta\epsilon_{kk_N} \delta_{ij} \tag{30}$$

Using the above equation to eliminate $\Delta\sigma_{ij}$ from eq. (16) yields:

$$[\Delta\sigma^t]_{N,K} = [\bar{D}^t]_{N,K-1} \, [\Delta\epsilon]_{N,K} \tag{31}$$

---

* An alternative interpretation of this method is to consider the undrained soil as incompressible, and to incorporate the incompressibility condition by means of a "penalty function" [22]. The associated "penalty number" corresponds to the bulk modulus Γ.

where (all components of [d] are zero, except $d_{11}=d_{22}=d_{33}=\Gamma$):

$$[\bar{D}^t]_{N,K-1} = [\bar{D}]_{N,K-1} + [d] \qquad (32)$$

A similar procedure is followed for eq. (18). It should be noted that eq. (31) is theoretically valid for all situations, including drained conditions ($\Gamma=0$). However, for very large values of Γ (undrained conditions), problems may arise if special precautions are not used [20].

2.3 Finite Element Applications - Properties Subroutine (CLAY)

A properties subroutine (FORTRAN 77) CLAY (and associated subroutines) has been prepared which evaluates the incremental stress-strain properties given by the bounding surface plasticity model for cohesive soils, i.e. the matrices $[D]_{N-1,K-1}$ and $[D]_{N,K-1}$ which appear in eqs. (17) and (18). A listing of the subroutine is provided in Appendix III. The subroutine is intended for incorporation into new or existing finite element programs for earth structures; such applications by the authors have proven to be straightforward and successful [12,14].

For each iteration of each increment, and for all points (e.g., element centers or quadrature points) in the body where the incremental properties are required, the parent finite element program calls subroutine CLAY. The call is as follows:

```
CALL CLAY (IDIM, INC, ITNO, ERMAX, PROP, STOR, SIGB, EPB, DSIG, DEP,
          DB, DE, UB, DLTAU, GAM, KIND, LARGE, LOCIT, TH1).
```

The quantities IDIM, INC, ITNO, KIND, LARGE and LOCIT are integer variables, ERMAX, UB, DLTAU, GAM, and TH1 are floating point variables, and PROP, STOR, SIGB, EPB, DSIG, DEP, DB and DE are floating point arrays of dimensions (19), (7), (6), (6), (6), (6), (6,6) and (6,6), respectively. With the

exception of IDIM, which is discussed in a later paragraph, the arguments in the call are described below:

INC: Increment number (the first increment must be numbered 1)

ITNO: Iteration number (the first iteration of each increment must be numbered 1)

ERMAX: Convergence limit for the local iteration, suggested values are .1 → .01.

PROP: An array containing the values of the material parameters which describe the bounding surface plasticity model for the soil at the point in the structure for which the incremental properties are sought. The parent finite element program must read and store the values of the soil parameters for each different type of soil in the earth structure, and then, for each call to CLAY, present the appropriate values for the element in question. The soil parameters are stored in the array in the following order (the significance of the various parameters are described in detail in [5,13]*; they are summarized in Appendix I): λ, κ, M_c, R_c, A_c, T, P_ℓ, ν or G, P_a, Γ, m, h_c, h_2, n, μ, r, a, c, s). That is, PROP(1)=λ, PROP(2)=κ, etc. At the time the properties are read into the main program, and before they are stored, M_c and P_ℓ must be multiplied by $\sqrt{3}/9$, and 3, respectively. It is suggested that subroutines RPROP and TCHECK, listed with EVAL in Appendix IV, be incorporated into the parent finite element program for the purpose of reading, echo printing and scaling the material parameters. The input formats for RPROP are those given in "III. Material Properties" of Appendix I.

STOR: This array is used to store certain quantities which vary with the current state of the soil (such as the current value of I_o) and thus,

---

* See beginning of the report for a discussion of modifications to the model.

are for a given step in the analysis unique to the point in the earth structure under consideration. The values in STOR must be stored (after each call to CLAY) by the parent finite element program for each point in the earth structure for which the incremental properties are needed (e.g., element centers). Prior to each call to CLAY the appropriate values for the point in question are retrieved from storage (i.e., from a two-dimensional array or a disk file which stores the values for each element in the system) and presented to the subroutine. At the beginning of the analysis the parent finite element program must initialize, for each point in question, STOR(1) and STOR(7), with the initial value of $3*P_o$ and e_o respectively. P_o and e_o are the initial values of the preconsolidation pressure ($I_o=3*P_o$ is the internal variable controlling the size of the bounding surface, Figure 1, [2]) and void ratio. These values are not read by subroutine RPROP, because in general they will vary from point to point in the deposit even though the soil is homogeneous, i.e., of one material type.

SIGB: $[\sigma]_{N-1}$; i.e., the <u>total</u> stress at the beginning of the increment; compressive stresses are taken to be positive.

EPB: $[\epsilon]_{N-1}$; the strain at the beginning of the increment; compressive strains are taken to be positive.

DSIG: $[\Delta\sigma]_{N,K-1}$; i.e., the estimate (supplied by the parent finite element program) of the <u>total</u> stress increment.*

DEP: $[\Delta\epsilon]_{N,K-1}$; i.e., the estimate (supplied by the parent finite element program) of the strain increment.*

DB & DE: $[D]_{N-1,K-1}$ or $[D^t]_{N-1,K-1}$ and $[D]_{N,K-1}$ or $[D^t]_{N,K-1}$ (see explanation for KIND); i.e., the estimates of the incremental stress-

---

* A discussion of the solution of initial estimates for these quantities is given in Appendix II.

strain properties for eqs. (17) and (18) calculated by the subroutine and supplied to the parent finite element program.

UB: u_{N-1}; i.e., the pore water pressure at the end of increment N-1 (see explanation for KIND).

DLTAU: $\Delta u_{N,K-1}$; i.e., the estimate of the pore water pressure increment (see explanation for KIND).

GAM: Γ; i.e., the combined bulk modulus for the soil particles and the pore water (see explanation for KIND).

KIND: A flag assigned a value of zero or one, depending on how undrained conditions are being modeled in the parent finite element program. A value of zero is required when the special formulation for incompressible and nearly-incompressible solids [18,20] is being used (i.e., the pore water pressure is treated as a primary dependent variable at the global level). In such cases the parent finite element analysis calculates u_{N-1} (UB) and $\Delta u_{N,K-1}$ (DLTAU) at the global level and supplies them to the subroutine. The value of Γ (GAM) is required at the global level for the nearly-incompressible formulation, and is supplied to the parent program by the subroutine. The DB array is the $[D]_{N-1,K-1}$ matrix of eqs. (17) and (18), etc.

A value of one is used for KIND when the conventional formulation for compressible solids is used for both drained and undrained conditions, [21]. (That is the only primary dependent variables are displacements, the pore water pressure is treated as a secondary dependent variable). In such cases the subroutine calculates the values of u_{N-1} (UB) and $\Delta u_{N,K-1}$ (DLTAU), and supplies them to the main program for printing purposes; the DB array is the $[\bar{D}^t]_{N-1,K-1}$ matrix of eq. (32), etc.

LARGE: A flag assigned a value of zero if engineering strains are used and a value of one if logarithmic (natural) strains are used.

LOCIT: Maximum number of local iterations per call, recommended values are 5-10.

TH1: Value of θ_1 used in eq. (17).

Subroutine CLAY computes three-dimensional incremental properties. The ordering of the stress and strain components in the $\{\sigma\}$ and $\{\epsilon\}$ vectors are indicated in eq. (10).

The subroutine can also be used to supply properties for two-dimensional finite element analyses. The procedure for its use in such cases and the value of the parameter IDIM in the subroutine call are described in the following paragraphs.

Axisymmetric Analysis (IDIM=3): The ordering of the stress and strain components are as follows $(\sigma_r,\sigma_\theta,\sigma_z,\tau_{r\theta}, 0.0,0.0)$ and $(\epsilon_r,\epsilon_\theta,\epsilon_z,\gamma_{r\theta},0.0,0.0)$. The indicated zero values must be supplied by the parent finite element program in the $\{\sigma\}_N$, $\{\epsilon\}_N$, $\{\Delta\sigma\}_{N,K-1}$ and $\{\Delta\epsilon\}_{N,K-1}$ vectors. The incremental properties of interest are in the upper-left 4 x 4 corners of the 6 x 6 DB and DE arrays returned by the subroutine.

Plane Stress (IDIM=3): The ordering of the stress and strain components are as follows $(\sigma_x,\sigma_y,0.0,\tau_{xy},0.0,0.0)$ and $(\epsilon_x,\epsilon_y,\epsilon_z,\gamma_{xy},0.0,0.0)$. The subroutine can only be used for supplying properties for plane stress finite element analyses that calculate the thickness strain, ϵ_z, at the global level. This limitation should not be of any consequence, because few earth structural problems are plane stress in nature.

Plane Strain: The subroutine can be used to supply properties for plane strain finite element analyses in two different ways. For plane strain programs which calculate the stress (σ_z) normal to the plane of the body, IDIM is given the

value of 3 and the ordering of the stress and strain vectors are $(\sigma_x,\sigma_y,\sigma_z,\tau_{xy},0.0,0.0)$ and $(\varepsilon_x,\varepsilon_y,0.0,\gamma_{xy},0.0,0.0)$, respectively. For plane strain analyses that do not calculate the value of σ_z, IDIM is given the value of 2 and the stress and strain vectors are $(\sigma_x,\sigma_y,\tau_{xy},0.0,0.0,0.0)$ and $(\varepsilon_x,\varepsilon_y,\gamma_{xy},0.0,0.0,0.0)$, respectively. The coefficients in the DB and DE arrays are appropriately arranged in each case.

2.4 Application to homogeneous tests

In the assessment of the characteristics of a material mode, and in the fitting of it to experimental measurements, a means must be available for using the model to predict the results of simple homogeneous tests. Program EVAL has been written for this purpose [10,17]. EVAL can be used for predicting the behavior of homogeneous soil samples subjected to arbitrary homogeneous stress and strain histories for either drained or undrained conditions.

The solution history is broken into "history segments." Within each history segment, a consistent combination of six stress and strain components are prescribed (i.e., the histories of ε_x or σ_x, ε_y or σ_y,...and γ_{yz} or τ_{yz}). A different combination may be prescribed in each history segment. For example, a uniaxial test might involve two segments with a specified value of axial strain achieved at the end of the first segment and with unloading to zero axial stress specified in the second. Each history segment is broken into increments with iteration conducted within each increment.

The analysis conducted by EVAL is essentially a one-element finite element analysis of a homogeneous body. For illustrative purposes the user can choose to perform either a conventional or "reformulated" analysis (see Section 2.3). When both analyses converge, they give identical results. For undrained and near-failure conditions, the reformulated analysis will, in certain cases, converge when the conventional analysis will not.

The reformulated analysis is a modification of the mixed finite element procedure reported in [18]. The strain components are augmented by the pore water pressure to form the "mixed" set of primary dependent variables. The set of governing equations are made up of the incremental effective stress-strain relations (eq. (16)) and the expression $0=\Gamma\ (\Delta\epsilon_x + \Delta\epsilon_y + \Delta\epsilon_z) - \Delta u$.

Because of the well behaved numerical characteristics of the model and the simplicity of the analysis for homogeneous tests, the method of successive approximation was found to be entirely adequate, i.e., $\theta_2=1/2$. The analysis is straightforward and well behaved with only two special features worth noting.

Because the analysis can be used for the extreme cases when either all the stress or all the strain components are specified, both the stress and strain vectors are checked for convergence. The convergence check on the stress increment, however, must be done with some care. The problem is that the relative measure of error, $L_1(\Delta\sigma_{N,K}-\Delta\sigma_{N,K-1})/L_1(\Delta\sigma_{N,K})$, used for the check is meaningless when applied for near failure conditions (because $\Delta\sigma_i \simeq 0$, $L_1(\Delta\sigma_{N,K})\simeq 0$). To avoid this difficulty, the denominator of the error measure is limited to a minimum value of $L_1(\sigma_{N-1})/10$. The L_1 norm is the sum of absolute values.

The second feature involves the starting estimates for the strain increment; the procedure outlined in Appendix II is used.

The "input" instructions for EVAL are given in Appendix I and a program listing in Appendix IV.

REFERENCES

1. Dafalias, Y.F., and L.R. Herrmann, "A Bounding Surface Soil Plasticity Model", Proceedings of the International Symposium of Soils Under Cyclic and Transient Loadings, University of Swansea, United Kingdom, pp. 335-345, January 1980.

2. Dafalias, Y.F., and L.R. Herrmann, "Bounding Surface Plasticity for Cohesive Soils", Final Report to the United States Army Engineer Waterways Experiment Station, Vicksburg, Mississippi (Re. Contract #DACA 39-79-M-0059), April 1980.

3. Dafalias, Y.F., L.R. Herrmann and J.S. DeNatale, "Prediction of the Response of the Natural Clays X and Y Using the Bounding Surface Model", Proceedings of the Workshop on Limit Equilibrium, Plasticity and Generalized Stress-Strain in Geotechnical Engineering, McGill University, ASCE, Part 1, pp. 402-415, May 1980.

4. Dafalias, Y.F., L.R. Herrmann and J.S. DeNatale, "Description of Natural Clay Behavior by a Simple Bounding Surface Plasticity Formulation", Proceedings of the Workshop on Limit Equilibrium, Plasticity and Generalized Stress-Strain in Geotechnical Engineering, McGill University, ASCE, Part 2, pp. 711-744, May 1980.

5. Herrmann, L.R., Y.F. Dafalias and J.S. DeNatale, "Bounding Surface Plasticity for Soil Modeling", Final Report to Civil Engineering Laboratory, Naval Construction Battalion Center, Port Hueneme, CA, Order No. USN N62583-80 M R478, October 1980.

6. Dafalias, Y.F., and L.R. Herrmann, "A Generalized Bounding Surface Constitutive Model for Clays", Proceedings of the Symposium on Limit Equilibrium, Plasticity and Generalized Stress Strain Applications in Geotechnical Engineering, ASCE, Hollywood, FL, pp. 78-95, October 1980.

7. Dafalias, Y.F., L.R. Herrmann and A. Anandarajah, "Cyclic Loading Response of Cohesive Soils Using a Bounding Surface Plasticity Model", Proceedings of the International Conference on Recent Advances in Geotechnical Earthquake Engineering and Soil Dynamics, University of Missouri-Rolla, Rolla, Missouri, Vol. 1, pp. 139-144, April 1981.

8. Herrmann, L.R., C.K. Shen, S. Jafroudi, J.S. DeNatale and Y.F. Dafalias, "A Verification Study for the Bounding Surface Plasticity Model for Cohesive Soils", Final Report to Civil Engineering Laboratory, Naval Construction Battalion Center, Port Hueneme, California, Order No. USN N62583-81MR320, December 1981.

9. Dafalias, Y.F., L.R. Herrmann and J.S. DeNatale, "The Bounding Surface Plasticity Model for Isotropic Cohesive Soils and its Application at the Grenoble Workshop", Genoble Workshop on Constitutive Relations for Soils, Sept. 6-8, Genoble, France, pp. 1-26, July 1982.

10. Herrmann, L.R., Y.F. Dafalias and J.S. DeNatale, "Numerical Implementation of a Bounding Surface Soil Plasticity Model", Proceedings of the International Symposium on Numerical Models in Geomechanics, Zurich, Switzerland, 1982.

11. Dafalias, Y.F., and L.R. Herrmann, "Bounding Surface Formulatin of Soil Plasticity", Chapter in Soil Mechanics - Transient and Cyclic Loads, John Wiley and Sons, Eds. O.C. Zienkiewicz and G.N. Pande, pp. 253-282, 1982.

12. Herrmann, L.R., J.S. DeNatale and Y.F. Dafalias, "Numerical Implementation of the Cohesive Soil Bounding Surface Plasticity Model (Volume I)" Civil Engineering, Naval Construction Battalion Center, Report CR83.010, February 1983.

13. DeNatale, J.S., L.R. Herrmann and Y.F. Dafalias, "User's Manual for MODCAL-Bounding Surface Soil Plasticity Model Calibration and Prediction Code (Volume II)," Civil Engineering, Naval Construction Battalion Center, Report CR83.011, February 1983.

14. Herrman, L.R., and K.D. Mish, "Finite Element Analysis for Cohesive Soil, Stress and Consolidation Problems Using Bounding Surface Plasticity Theory", Department of Civil Engineering, University of California, Davis Report to: Civil Engineering Laboratory Naval Construction Battalion Center, Port Hueneme, California, Order No. N62583-83-M-T062, September 1983.

15. Dafalias, Y.F., "Bounding Surface Elastoplasticity-Viscoplasticity for Particulate Cohesive Media", IUTAM Symposium on Deformation and Failure of Granular Materials, Delft, The Netherlands, August 1982.

16. Owen, D.R.J., and E. Hinton, "Finite Elements in Plasticity - Theory and Practice, Pineridge Press, Swansea, U.K., 1980.

17. Herrmann, L.R., and M.A. Taylor, "Characterization of the Structural Behavior of Rock Masses", Final Report to the U.S. Bureau of Mines, Contract #G0133122, Vol. I&II, September 1974.

18. Herrmann, L.R., "Elasticity Equations for Incompressible and Nearly Incompressible Materials by a Variational Theorem", AIAA J., Vol. 3, No. 10, 1896-1900, 1965.

19. Taylor, R.L., K.S. Pister and L.R. Herrmann, "On a Variational Theorem for Incompressible and Nearly Incompressible Orthotropic Elasticity", Int. J. of Solids and Structures, Vol. 4, 875-883, 1968.

20. Zienkiewicz, O.C., R.L. Taylor and J.M.W. Baynham, "Mixed and Irreducible Formulations in Finite Element Analysis", Proceedings International Symposium on Hybrid and Mixed Finite Element Methods, Atlanta, GA, April 1981.

21. Sangrey, D.A., D.J. Henkel and M.I. Espig, "The Effective Stress Response of a Saturated Clay Soil to Repeated Loading", Canadian Geotechnical Journal, 1969, 6(3), 241-252.

22. Zienkiewicz, O.C., The Finite ELement Method, McGraw-Hill, Ltd., London, 1979.

23. Naylor, D.J., "Stresses in Nearly Incompressible Materials for Finite Element with Application to the Calculations of Excess Pore Pressures", Int. J. Num. Meth. Eng., 8, pp. 443-460, 1974.

24. Nagtegaal, J.C., D.M. Parks and J.R. Rice, "On Numerically Accurate Finite Element Solutions in the Fully Plastic Range", Comp. Meth. Appl. Mech. Eng., 4, pp. 153-178, 1974.

Appendix I: Input Instructions for EVAL

I. Heading information

Line 1 (20A2)

Columns

1 - 80	ITITLE	:	Information that is to be printed as a title for the analysis

II. Initial State Parameters

Line 1 (3E10.3)

Columns

1 - 10	P_{o_i}	:	Initial value of effective preconsolidation pressure
11 - 20	e_o	:	Initial void ratio
21 - 30	σ_{c_o}	:	Initial hydrostatic confining pressure*

---

* The strains produced by the application of σ_{c_o} are not calculated. It is assumed that σ_{c_o} is applied under drained conditions (regardless of value of Γ). If it is desired to calculate the strains due to σ_{c_o} and/or to apply σ_{c_o} under undrained conditions, then σ_{c_o} is set equal to zero and the confining pressure is applied in history segment 1.

III. Material Parameters*

Line 1 (8E10.3)

Columns

1 - 10	λ	: Slope of isotropic consolidation line for an e-ℓn p′ plot
11 - 20	κ	: Slope of elastic rebound line for an e-ℓn p′ plot
21 - 30	M_c	: Slope of critical state line in triaxial space (compression)
31 - 40	R_c	: Parameters describing shape of bounding surface (compression)
41 - 50	A_c	
51 - 60	T	
61 - 70	p_ℓ	: Transitional value of confining pressure separating linear rebound curves on e-ℓn p′ and e-p′ plots. Suggested range of values = $.3P_a \rightarrow 1.0P_a$
71 - 80	ν or G	: Poisson's ratio or shear modulus

Line 2 (8E10.3)

Columns

1 - 10	P_a	: Atmospheric pressure
11 - 20	Γ	: Combined bulk modulus for soil particles and pore water (Γ=0 for drained conditions); if no information is available, it is suggested that a value of 20,000 P_a be used
21 - 30	m	: Hardening parameter
31 - 40	h_c	: Shape hardening parameter for compression
41 - 50	h_2	: Shape hardening parameter on the I-axis
51 - 60	$n=M_e/M_c$	: Ratio of extension to compression values
61 - 70	$\mu=h_e/h_c$	
71 - 80	$r=R_e/R_c$	

---

* Note: The input in this group is read by subroutine RPROP, detailed definitions of the several material parameters are given in [5,13].

Line 3 (3E10.3)

Columns		
1 - 10	$a=A_e/A_c$	: Ratio of extension to compression values
11 - 20	C	: Projection center variable
21 - 30	s	: Elastic zone variable (a value of 1.0 gives no elastic zone)

IV. Iteration Information

Line 1 (I5,2E10.2,3I5)

Columns		
1 - 5	ITMAX	: Max. no. of iterations per increment (values of 5-10 are suggested)
6 - 15	ERMAX	: Maximum permitted relative difference $\Delta L_K/L_K$ for the norms of the incremental stress and strain vectors. Values of .01 to .001 are suggested. If convergence does not occur in ITMAX iterations, the program prints a message and then continues to the next problem
16 - 25	CONFL	: $(0.0 < (\ ) \leq 1.0)$ Establishes upper and lower limits of $\frac{1}{\text{CONFL}}$ and CONFL for the calculated values of the Aitken's acceleration factors (if it is desired not to use acceleration factors set CONFL = 1.0).
26 - 30	KIND	: 0 Reformulated (for nearly-incompressible conditions) analysis 1 Conventional analysis
31 - 35	LARGE	: 0 Engineering stresses and strains used in analysis 1 True stresses and strains used in analysis
36 - 40	LOCIT	: Maximum number of iterations to be used locally within subroutine CLAY (default=1)

V. Output Control Information

Line 1 (2I5)

Columns

1 - 5	IPRNT1	=	0 print incremental stress and strain values 1 suppress printing of incremental values
6 - 10	IPRNT2	=	0 print computed incremental critical state stresses and strains 1 suppress printing of incremental critical state values

VI. Printer Plot Control Information

Line 1 10(4X,I1)

Columns

5I	IPLOT (I)	=	0 Do not generate printer plot type I* 1 Generate printer plot type I*

VII. Description of M history segments

For each of the M history segments, one line (6(I1,E9.3), I5, E10.3) is required:

Columns

1 $\quad ICOD_1 = \begin{cases} 0 - \sigma_{x_M} \\ 1 - \varepsilon_{x_M} \end{cases}$** is specified

2 - 10 $\quad V_1 =$ value of $\begin{cases} \sigma_{x_M} \\ \varepsilon_{x_M} \end{cases}$ for $IC_1 = \begin{cases} 0 \\ 1 \end{cases}$

---

* For I=1,6 — for an explanation of the contents of plot "I" see note following "experimental data for plotting" (section IX).

** σ_{x_M} is the value of σ_x at the end of segment M, etc. Thus, when $IC_1 = 0$, a change in σ_x of $(\sigma_{x_M} - \sigma_{x_{M-1}})$ is applied in NINC increments during the loading segment. $\sigma_{x_{M-1}}$ is the value of σ_x calculated ($IC_{1_{M-1}} = 1$) or specified ($IC_{1_{M-1}} = 0$) at the end of segment M-1.

$$11 \qquad ICOD_2 = \begin{Bmatrix} 0 - \sigma_{y_M} \\ 1 - \varepsilon_{y_M} \end{Bmatrix} \text{ is specified}$$

$$12 - 20 \qquad V_2 = \text{value of } \begin{Bmatrix} \sigma_{y_M} \\ \varepsilon_{y_M} \end{Bmatrix} \text{ for } IC_2 = \begin{cases} 0 \\ 1 \end{cases}$$

$$21 \qquad ICOD_3 = \begin{Bmatrix} 0 - \sigma_{z_M} \\ 1 - \varepsilon_{z_M} \end{Bmatrix} \text{ is specified}$$

$$22 - 30 \qquad V_3 = \text{value of } \begin{Bmatrix} \sigma_{z_M} \\ \varepsilon_{z_M} \end{Bmatrix} \text{ for } IC_3 = \begin{cases} 0 \\ 1 \end{cases}$$

$$31 \qquad ICOD_4 = \begin{Bmatrix} 0 - \tau_{xy_M} \\ 1 - \gamma_{xy_M} \end{Bmatrix} \text{ is specified}$$

$$32 - 40 \qquad V_4 = \text{value of } \begin{Bmatrix} \tau_{xy_M} \\ \gamma_{xy_M} \end{Bmatrix} \text{ for } IC_4 = \begin{cases} 0 \\ 1 \end{cases}$$

$$41 \qquad ICOD_5 = \begin{Bmatrix} 0 - \tau_{xz_M} \\ 1 - \gamma_{xz_M} \end{Bmatrix} \text{ is specified}$$

$$42 - 50 \qquad V_5 = \text{value of } \begin{Bmatrix} \tau_{xz_M} \\ \gamma_{xz_M} \end{Bmatrix} \text{ for } IC_5 = \begin{cases} 0 \\ 1 \end{cases}$$

$$51 \qquad ICOD_6 = \begin{Bmatrix} 0 - \tau_{yz_M} \\ 1 - \gamma_{yz_M} \end{Bmatrix} \text{ is specified}$$

$$52 - 60 \qquad V_6 = \text{value of } \begin{Bmatrix} \tau_{yz_M} \\ \gamma_{yz_M} \end{Bmatrix} \text{ for } IC_6 = \begin{cases} 0 \\ 1 \end{cases}$$

61 - 65	NINC	=	number of increments into which segment M is to be divided
66 - 75	SR	=	Increment ratio for specified quantities (e.g., if $IC_1 = 0$ then $\Delta\sigma_{x_N} / \Delta\sigma_{x_{N-1}} =$ SR; $\Delta\sigma_{x_N}$ is the increment of σ_x applied during increment N of loading segment M, etc.). A value of 1.0 gives equal magnitude increments for the segment.

VIII. Termination of Loading History Input

Line 1 (I1)

Column

1	:	enter the integer 9 to signify the end of *loading history input*

IX. Experimental data for plotting

For each of the "printer plots" *requested on the "printer plot control line"* the following information must be specified.

Line 1 (I5)

Columns

1 - 5	NE	=	Number of experimental points to be *printer plotted* (symbol #). The calculated points will likewise be printer plotted (symbol *).

Lines 2-(N+1) (8E10.3), where N = NE/8

Columns

1 - 10	x_{e_1}	Lines 2 contains the abscissas of the first eight experimental points, etc.
11 - 20	x_{e_2}	
21 - 30	x_{e_3}	
31 - 40	x_{e_4}	
41 - 50	x_{e_5}	
51 - 60	x_{e_6}	
61 - 70	x_{e_7}	
71 - 80	x_{e_8}	

Line (N+2) - (2N+1) (8E10.3)

Columns	
1 - 10	y_{e_1}
11 - 20	y_{e_2}
21 - 30	y_{e_3}
31 - 40	y_{e_4}
41 - 50	y_{e_5}
51 - 60	y_{e_6}
61 - 70	y_{e_7}
71 - 80	y_{e_8}

Line (N+2) contains the ordinates of the first eight experimental points, etc.

Note: Currently 6 plotting slots (the maximum number of plots can easily be increased) are in use (the items being plotted in a given plot can easily be changed), i.e.

I = 1: $Q = (\sigma_z - \sigma_x)$ vs. $P = ((\sigma_x + \sigma_y + \sigma_z)/3 - u)$

I = 2: Q vs. ε_z

I = 3: u vs. ε_z

I = 4: $\Delta V/V_o$ (volume change) vs. ε_z

I = 5: Q/P_o vs. P/P_o

I = 6: Q/P_o vs. ε_z

The above input sequence (sections I-IX) is repeated for each subsequent analysis.

Appendix II: Initial Estimates for the Stress and Strain Increments

The matrices $[D]_{N-1,K-1}$ and $[D]_{N,K-1}$ are calculated using the K-1 estimates of the stress and strain vectors, thus at the beginning of the iteration process, initial estimates are required. For the first iteration of the first increment, they are usually taken to be zero. For the intial iterations of succeeding increments, they also can be started at zero; however, it is usually desirable to make use of information from the previous increment to obtain better starting values. The simplest procedure is to use as the intial estimate the final values found in the previous increment multiplied by the ratio of the time increments involved. This practice is based on the assumption of relatively uniform behavior from increment to increment. Difficulties can arise when the histories of applied external loads or displacements acting on the structure cause a switch from loading to unloading in an unstable material response regime. For example, consider the one-dimensional response shown in Figure 3. Consider the case when the state of the soil is at point "A" at the end of increment N-1. If during increment N, $\Delta\sigma_N$ is specified, two final states B and B' are possible. One corresponds to $\Delta\varepsilon$ (negative) and the other to $\Delta\varepsilon'$ (positive). Without any additional information, no choice can be made between B and B'. (It is easily seen that for specified stress increments in the stable region of behavior and for specified strain increments anywhere, no such problems exists.) The suggested solution to this impasse is to assume that the user would not attempt a stress controlled specification for "loading" conditions (path A-B') in an unstable region and, hence, if the stress increment is specified, unloading is the proper behavior (path A-B)[†]. For stress controlled conditions, the selection of the unloading path can be assured if the starting estimate of strain is of opposite sign to that calculated in the previous increment. Thus, the following

---

[†] This argument requires that the arrival at A must have involved strain controlled steps.

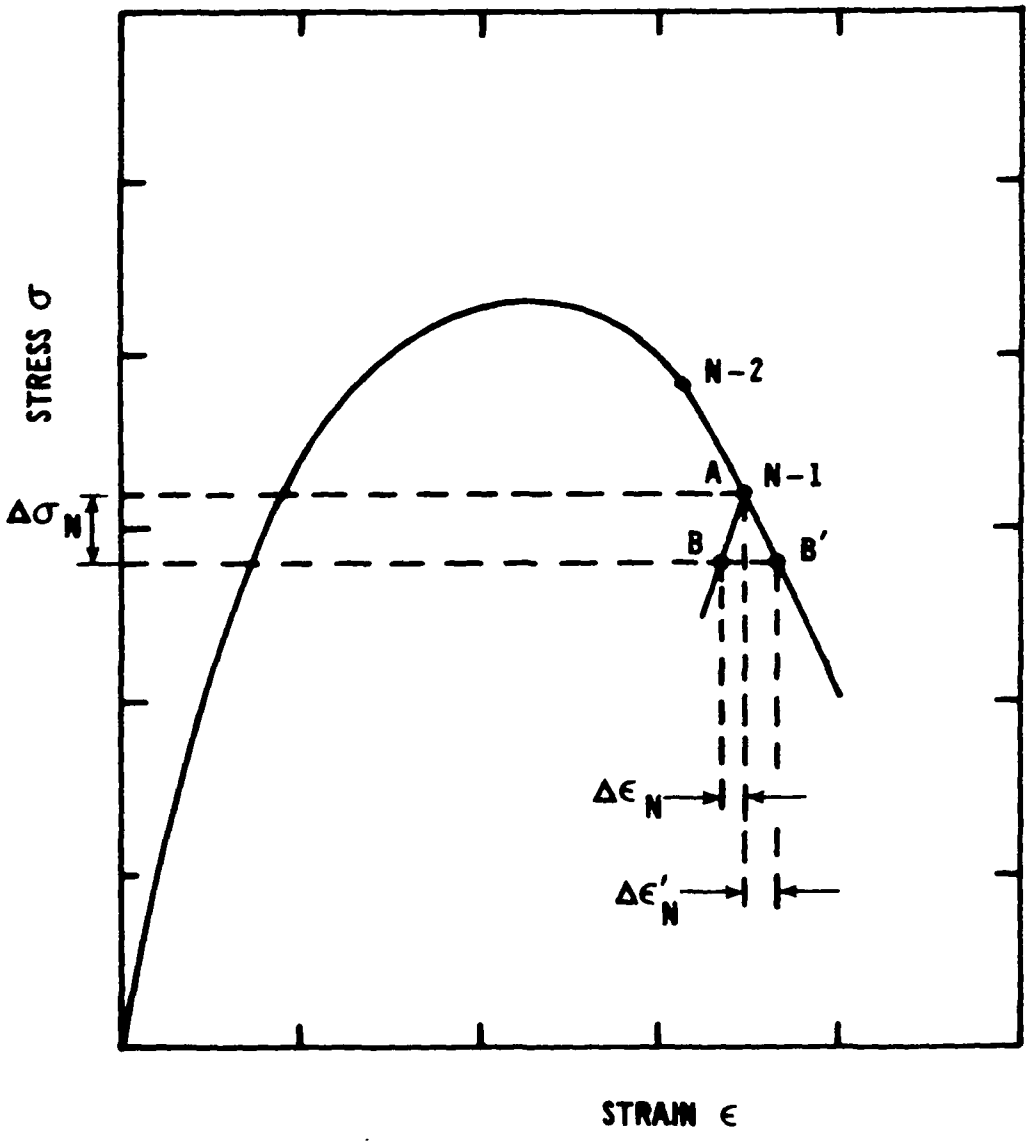


Figure 3. One-Dimensional Soil Response

strategy is recommended. When considering a series of increments for which the rates of the externally applied loads and displacements do not change sign[††], $\{\Delta\varepsilon\}_{N-1}$ and $\{\Delta\sigma\}_{N-1}$ are used as starting estimates for increment N. However, as one such solution history segment is ended and a new one begins, the prerequisite conditions for the non-uniqueness problem may occur. Hence, for the first increment of each such series, it is suggested that the starting strain estimate be taken as some small negative multiple (e.g. -.01) of the value found in the previous increment (the stress increment would be used unchanged). The reduction in absolute magnitude is in deference to the greater stiffness encountered in unloading. Such an initial estimate will force the solution to select path A → B if the necessary conditions exist for the non-uniqueness to occur. If non-uniqueness is not a problem, the only effect of this procedure is to slightly slow the convergence process.

[††] It is assumed that this condition is sufficient to prevent a general switch from loading to unloading within the soil mass.

Appendix III: Listing of Properties Subroutine CLAY

```
      SUBROUTINE CLAY (IDIM,INC,ITNO,ERMAX,PROP,STOR,SIGBM,EPM,
     *          DSIGM,DEPM,DB,DE,UB,DLTAU,GAMMA,KIND,LARGE,LOCIT,TH1)
C ***************************************************************************
C *      Subroutine to evaluate Yannis Dafalias' bounding                   *
C *           surface plasticity model for clay soils.                      *
C *    Fortran 77 version, Prepared by L.R. Herrmann and V.Kaliakin         *
C *         at the University of California, Davis Campus.                  *
C ***************************************************************************
      INTEGER I,J,K,IT,IDIM,INC,ITNO,KIND,LARGE,LOCIT,II(6)
      REAL PROP(19),STOR(7),SIGBM(6),EPM(6),DSIGM(6),DEPM(6),DB(6,6),
     *     DE(6,6),SIGB(6),EPB(6),DSIG(6),DEP(6),DEPT(3,3),SB(3,3),
     *     SE(3,3),DLTA(3,3),ERMAX,UB,DLTAU,GAMMA,GAMMAT,SMALL,DIL,
     *     DDIL,VOIDB,VOIDE,XIB,XIE,XJB,XJE,SCUBEB,SCUBEE,SIN3AB,
     *     SIN3AE,COS3AB,COS3AE,BULKB,BULKE,GB,GE,XIOB,XIOE,XIL,
     *     TH1,TEMP1,TEMP2,TEMP3,TEMP4
C
      DATA II/11,22,33,12,13,23/, DLTA/1.0,3*0.0,1.0,3*0.0,1.0/
      SMALL=0.0001*PROP(8)
C
      DO 100 I=1,6
         SIGB(I)=SIGBM(I)
         DSIG(I)=DSIGM(I)
         EPB(I)=EPM(I)
         DEP(I)=DEPM(I)
  100 CONTINUE
C
C        Initialize history if necessary
C
      IF(INC .EQ. 1 .AND. ITNO .EQ. 1)  THEN
          STOR(2)=STOR(1)
          STOR(3)=0.0
          STOR(5)=0.5*(SIGB(1) + SIGB(2))
          STOR(6)=0.01*PROP(8)
      ELSE
C
C        Update history if necessary
C
          IF(INC .GT. 1 .AND. ITNO .EQ. 1)  THEN
              STOR(1)=STOR(2)
              STOR(3)=STOR(3) + STOR(4)
              STOR(5)=STOR(5) + STOR(6)
           END IF
      END IF
C
C    Convert from plane strain to 3-dimensonal state if necessary
C
      IF(IDIM .EQ. 2)
     *          CALL TWODIM (1,SIGB,EPB,DSIG,DEP,DB,DE,STOR)
C
      DIL=0.0
      DDIL=0.0
      DO 200 I=1,3
          DIL =DIL  +  EPB(I)
          DDIL=DDIL +  DEP(I)
  200 CONTINUE
C
C       Determine 3-dimensional incremental properties.
C            Iterate on the stress estimate.
C
      DO 600 IT=1,LOCIT
```

```
C
C               Compute values of the invariants
C
      IF(IT .EQ. 1)
     *    CALL INVAR (1,PROP,STOR,SIGB,DSIG,DEP,DEPT,VOIDB,KIND,
     *      LARGE,SMALL,XIB,XJB,DIL,DDIL,SB,SCUBEB,SIN3AB,COS3AB,
     *      GAMMA,GAMMAT,UB,DLTAU,II,DLTA)
      CALL INVAR (2,PROP,STOR,SIGB,DSIG,DEP,DEPT,VOIDE,KIND,
     *      LARGE,SMALL,XIE,XJE,DIL,DDIL,SE,SCUBEE,SIN3AE,COS3AE,
     *      GAMMA,GAMMAT,UB,DLTAU,II,DLTA)
C
C            Calculate elastic incremental properties
C
      IF(IT .EQ. 1)
     *    CALL ELASTC (PROP,VOIDB,XIB,DB,BULKB,GB,GAMMAT,II,DLTA)
      CALL ELASTC (PROP,VOIDE,XIE,DE,BULKE,GE,GAMMAT,II,DLTA)
C
C           Calculate the size of bounding surface
C
      XIOB=STOR(1)
      XIOE=STOR(2)
      XIL=PROP(7)
      TEMP1=1.0/(PROP(1) - PROP(2))
      TEMP2=(XIE - XIB)/3.0
      IF(XIOB .GE. XIL .AND. XIOE .GE. XIL)  THEN
          XIOE=XIOB*EXP(TEMP1*0.5*((VOIDB + VOIDE)*DDIL
     *          - (VOIDB/BULKB + VOIDE/BULKE)*TEMP2))
      ELSE
          TEMP3=XIOB
          IF(XIOB .LT. XIL)  TEMP3=XIL
          TEMP4=XIOE
          IF(XIOE .LT. XIL)  TEMP4=XIL
          XIOE=XIOB + TEMP1*0.5*((TEMP4*VOIDE+TEMP3*VOIDB)*DDIL
     *       - (TEMP3*VOIDB/BULKB + TEMP4*VOIDE/BULKE)*TEMP2)
      END IF
      STOR(2)=XIOE
C
      IF(INC + ITNO + IT .GT. 3)  THEN
C
C     Calculate the bounding surface parameters and parameters associated
C           with the plastic portion of the incremental properties.
C
          IF(IT .EQ. 1 .OR. (INC + ITNO + IT .EQ. 4))
     *        CALL PLASTC (PROP,DEPT,VOIDB,XIB,XJB,XIOB,DDIL,
     *              SB,SCUBEB,SIN3AB,COS3AB,DB,BULKB,GB,II,DLTA)
          CALL PLASTC (PROP,DEPT,VOIDE,XIE,XJE,XIOE,DDIL,SE,SCUBEE,
     *              SIN3AE,COS3AE,DE,BULKE,GE,II,DLTA)
C
C       Calculate revised total stress estimates and error norms
C
          IF(IDIM .EQ. 2)
     *        CALL TWODIM (2,SIGB,EPB,DSIG,DEP,DB,DE,STOR)
          TEMP2=0.0
          TEMP3=0.0
          TEMP4=0.0
          DO 400 I=1,6
              J=II(I)/10
              K=MOD(II(I),10)
              TEMP1=DLTA(K,J)*DLTAU
              DO 300 J=1,6
```

```
                    TEMP1=TEMP1 + ((1.0 - TH1)*DB(I,J)
     *                          + TH1*DE(I,J))*DEP(J)
  300             CONTINUE
                  TEMP2=TEMP2 + ABS(TEMP1 - DSIG(I))
                  TEMP3=TEMP3 + ABS(TEMP1)
                  TEMP4=TEMP4 + ABS(SIGB(I))
                  DSIG(I)=TEMP1
  400         CONTINUE
              IF(TEMP3 .LT. TEMP4*0.01)  TEMP3=0.01*TEMP4
              IF(TEMP3 .NE. 0.0)  THEN
                  IF(TEMP2/TEMP3 .LT. ERMAX)  GOTO 700
              END IF
          END IF
  600 CONTINUE
  700 CONTINUE
C
C      Convert 3-dimensional properties to plane strain if necessary
C
      IF(IDIM .EQ. 2)
     *    CALL TWODIM (3,SIGB,EPB,DSIG,DEP,DB,DE,STOR)
      RETURN
      END
```

```
      SUBROUTINE INVAR (IK,PROP,STOR,SIG,DSIG,DEP,DEPT,VOID,KIND,
     *                  LARGE,SMALL,XI,XJ,DIL,DDIL,S,SCUBE,SIN3A,COS3A,
     *                  GAMMA,GAMMAT,UB,DLTAU,II,DLTA)
C
C             Subroutine to compute values of invariants
C
      INTEGER I,J,K,N,IK,KIND,LARGE,II(6)
      REAL PROP(19),STOR(7),SIG(6),DSIG(6),DEP(6),DEPT(3,3),S(3,3),
     *     DLTA(3,3),VOID,SMALL,XI,XJ,DIL,DDIL,SCUBE,SIN3A,COS3A,GAMMA,
     *     GAMMAT,UB,DLTAU,ARB,FACTOR,TEMP1
C
      DATA ARB/1000.0/
      FACTOR=0.0
      IF(IK .GT. 1)  FACTOR=1.0
      XI =0.0
      XJ =0.0
      SCUBE=0.0
      SIN3A=0.0
C
C        Calculate first stress invariant
C
      DO 100 I=1,3
          XI=XI + SIG(I) + FACTOR*DSIG(I)
  100 CONTINUE
C
      VOID=1.0 + STOR(7)
      IF(LARGE .NE. 0)
     *         VOID=VOID*EXP(-DIL - FACTOR*DDIL)
C
C      Change tensor components to matrix components,
C             calculate deviatoric stresses.
C
      DO 200 N=1,6
         I=II(N)/10
         J=MOD(II(N),10)
         S(I,J)=SIG(N) + FACTOR*DSIG(N) - XI*DLTA(I,J)/3.0
         S(J,I)=S(I,J)
         DEPT(I,J)=DEP(N)*(1.0 + DLTA(I,J))*0.5
         DEPT(J,I)=DEPT(I,J)
  200 CONTINUE
C
C        Convert total stresses to effective stresses
C
      GAMMA=PROP(6)
      GAMMAT=0.0
      IF(KIND .NE. 0) THEN
          GAMMAT=GAMMA
          UB=STOR(3)
          DLTAU=GAMMA*DDIL
      END IF
      XI=XI - 3.0*(UB + FACTOR*DLTAU)
      STOR(4)=DLTAU
C
C      Avoid near zero value of the first stress invariant
C
      IF(ABS(XI) .LE. SMALL) THEN
          TEMP1=XI
          XI=SMALL
          IF(TEMP1 .LT. 0.0)  XI=-SMALL
      END IF
```

```
C
C    Compute the square root of the second deviatoric stress invariant
C           as well as the third deviatoric stress invariant
C
      DO 300 I=1,3
          DO 300 J=1,3
              XJ=XJ + S(I,J)*S(I,J)
              DO 300 K=1,3
                  SCUBE=SCUBE + S(I,J)*S(J,K)*S(K,I)
  300 CONTINUE
      SCUBE=SCUBE/3.0
C
C      Arbitrary check to avoid excessively small values of J
C
      XJ=SQRT(0.5*XJ)
      IF(XJ*ARB .LT. XI)  XJ=0.0
C
C      Compute the sine and cosine of three times the "Lode" angle
C
      IF(XJ .GT. SMALL)  SIN3A=1.5*SQRT(3.0)*SCUBE/XJ**3
      IF(SIN3A .GT.  1.0)  SIN3A= 1.0
      IF(SIN3A .LT. -1.0)  SIN3A=-1.0
      COS3A=SQRT(1.0 - SIN3A**2)
C
      RETURN
      END
```

```
      SUBROUTINE ELASTC (PROP,VOID,XI,D,BULK,G,GAMMAT,II,DLTA)
C
      INTEGER I,J,K,L,M,N,II(6)
      REAL PROP(19),D(6,6),DLTA(3,3),VOID,XI,BULK,G,GAMMAT,TEMP1,
     *     TEMP2,TEMP3
C
C        Calculate the bulk and shear moduli
C
      TEMP1=1.5*(1.0 - 2.0*PROP(5))/(1.0 + PROP(5))
      TEMP2=VOID/3.0/PROP(2)
      TEMP3=XI
      IF(TEMP3 .LT. PROP(7))  TEMP3=PROP(7)
      BULK=TEMP2*TEMP3
      IF(PROP(5) .LE. 0.5) THEN
          G=TEMP1 * BULK
      ELSE
          G=PROP(5)
      END IF
C
C        Calculate elastic incremental properties
C
      DO 100 M=1,6
         I=II(M)/10
         J=MOD(II(M),10)
         DO 100 N=M,6
             K=II(N)/10
             L=MOD(II(N),10)
             TEMP1=DLTA(K,I)*DLTA(L,J) + DLTA(K,J)*DLTA(I,L)
             D(M,N)=TEMP1*G + (BULK + GAMMAT
     *                - 2.0*G/3.0)*DLTA(I,J)*DLTA(K,L)
             D(N,M)=D(M,N)
  100 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE PLASTC (PROP,DEPT,VOID,XI,XJ,XIO,DDIL,S,SCUBE,
     *                   SIN3A,COS3A,D,BULK,G,II,DLTA)
C
      INTEGER I,J,K,L,M,N,LL,LFLAG,II(6)
      REAL PROP(19),DEPT(3,3),S(3,3),D(6,6),DLTA(3,3),VOID,X,XI,
     *     XJ,XIO,DDIL,SCUBE,SIN3A,COS3A,BULK,G,XKP,XKPB,BETA,
     *     DBETA,DFI,DFJ,DFAL,DFJJ,TEMP,TEMP1,TEMP2,TEMP3,TEMP4,
     *     TEMP5,TEMP6
C
C             Calculate bounding surface parameters
C
      CALL BOUND (PROP,VOID,X,XI,XJ,XIO,SIN3A,XKPB,BETA,
     *          DFI,DFJ,DFAL,DFJJ)
      DBETA=BETA - 1.0
      IF(DBETA .LT. 0.0)  DBETA=0.0
C
C             Check for elastic zone
C
      LFLAG=0
      TEMP1=BETA - DBETA*PROP(15)
      IF(TEMP1 .GT. 0.0)  THEN
          LFLAG=1
C
C         Calculate the plastic modulus and loading function
C
          CALL LODFUN (PROP,DEPT,VOID,X,XJ,XIO,DDIL,S,SCUBE,SIN3A,
     *       COS3A,BULK,G,XKP,XKPB,DBETA,TEMP1,DFI,DFJ,DFAL,DFJJ,LFLAG)
      END IF
C
C       Calculate plastic portion of the incremental properties
C
      IF(LFLAG .NE. 0)  THEN
          DO 200 M=1,6
              I=II(M)/10
              J=MOD(II(M),10)
              DO 200 N=M,6
                  K=II(N)/10
                  L=MOD(II(N),10)
C
                  TEMP=0.0
                  TEMP1=0.0
                  TEMP2=0.0
                  TEMP3=3.0*BULK*DFI
                  TEMP4=G*DFJJ
                  TEMP5=SQRT(3.0)*G*DFAL
                  TEMP6=XKP + 9.0*BULK*DFI*DFI + G*DFJ*DFJ
     *                + G*(DFAL*COS3A)*(DFAL*COS3A)
                  IF(XJ**4 .NE. 0.0)  THEN
                      DO 100 LL=1,3
                          TEMP1=TEMP1 + S(I,LL)*S(LL,J)
                          TEMP2=TEMP2 + S(K,LL)*S(LL,L)
  100                 CONTINUE
                      TEMP1=TEMP5*(TEMP1/XJ**2 - 1.5*SCUBE*S(I,J)/XJ**4
     *                    -2.0*DLTA(I,J)/3.0)
                      TEMP2=TEMP5*(TEMP2/XJ**2 - 1.5*SCUBE*S(K,L)/XJ**4
     *                    -2.0*DLTA(K,L)/3.0)
                  END IF
                  TEMP=(TEMP3*DLTA(I,J) + TEMP4*S(I,J) + TEMP1)
     *                *(TEMP3*DLTA(K,L) + TEMP4*S(K,L) + TEMP2)/TEMP6
                  D(M,N)=D(M,N) - TEMP
```

```
                D(N,M)=D(M,N)
200     CONTINUE
    END IF
    RETURN
    END
```

```
      SUBROUTINE LODFUN (PROP,DEPT,VOID,X,XJ,XIO,DDIL,S,SCUBE,SIN3A,
     *            COS3A,BULK,G,XKP,XKPB,DBETA,DDEN,DFI,DFJ,DFAL,DFJJ,LF)
C
C      Subroutine to calculate the plastic modulus and loading function
C
      INTEGER I,J,K,LF
      REAL ALFUN,CV,RT,SINV
      REAL PROP(19),DEPT(3,3),S(3,3),VOID,X,XJ,XIO,DDIL,SCUBE,SIN3A,
     *     COS3A,BULK,G,XKP,XKPB,DBETA,DDEN,DFI,DFJ,DFAL,DFJJ,XN,R,
     *     Z,XIL,XM,H1,H2,H,SUM,XLF,TEMP1,TEMP2,TEMP3,TEMP4,TEMP5
      ALFUN(CV,RT,SINV)=2.0*RT*CV/(1.0 + RT - (1.0 - RT)*SINV)
      XN=ALFUN(PROP(3), PROP(4), SIN3A)
      R =ALFUN(PROP(9), PROP(12),SIN3A)
      H1=ALFUN(PROP(16),PROP(18),SIN3A)
      Z=XJ*R/(XN*XIO)
      XIL=PROP(7)
      XM=PROP(17)
      H2=PROP(19)
C
C                    Calculate the plastic modulus
C   (using modified formulation for continuity across the I-axis)
C
      TEMP1=1.0/(PROP(1) - PROP(2))
      TEMP2=Z**XM
      TEMP3=9.0*DFI*DFI + DFJ*DFJ/3.0
      TEMP4=XIO
      IF(XIO .LT. XIL)  TEMP4=XIL
      H=H1*TEMP2 + H2*(1.0 - TEMP2)
      XKP=XKPB + H*DBETA/DDEN*TEMP1*VOID*TEMP3*TEMP4
C
C         Calculate the loading function
C
      SUM=0.0
      TEMP1=0.0
      TEMP2=3.0*BULK*DFI
      TEMP3=G*DFJJ
      TEMP4=SQRT(3.0)*G*DFAL
      TEMP5=XKP + 9.0*BULK*DFI*DFI + G*DFJ*DFJ
     *               + G*(DFAL*COS3A)*(DFAL*COS3A)
      IF(XJ**2 .NE. 0.0)  THEN
          DO 200 I=1,3
          DO 200 J=1,3
              TEMP2=0.0
              DO 100 K=1,3
                  TEMP2=TEMP2 + S(I,K)*S(K,J)
  100         CONTINUE
              TEMP1=TEMP1 + (TEMP2 - 1.5*SCUBE*S(I,J)/(XJ*XJ))
     *              *DEPT(I,J)/(XJ*XJ)
              SUM=SUM + S(I,J)*DEPT(I,J)
  200     CONTINUE
          TEMP1=TEMP1 - 2.0*DDIL/3.0
      END IF
      XLF=(TEMP2*DDIL + TEMP3*SUM + TEMP4*TEMP1)/TEMP5
C
C           Check for unloading
C
      IF(XLF .LE. 0.0)  LF=0
      RETURN
      END
```

```
      SUBROUTINE TWODIM (IK,SIGB,EPB,DSIG,DEP,DB,DE,STOR)
C
C         Subroutine to perform manipulation of storage locations
C                   for the case of plane strain.
C
      INTEGER I,IK
      REAL SIGB(6),EPB(6),DSIG(6),DEP(6),DB(6,6),DE(6,6),STOR(7),TEMP1
C
      IF(IK .EQ. 1)  THEN
          SIGB(4)=SIGB(3)
          SIGB(3)=STOR(5)
          DSIG(4)=DSIG(3)
          DSIG(3)=STOR(6)
          EPB(4)=EPB(3)
          EPB(3)=0.0
          DEP(4)=DEP(3)
          DEP(3)=0.0
          DO 100 I=5,6
              SIGB(I)=0.0
              DSIG(I)=0.0
              EPB(I) =0.0
              DEP(I) =0.0
  100     CONTINUE
      ELSE IF(IK .EQ. 2)  THEN
C
C       Compute and store the stress increment in the Z-direction
C
          TEMP1=0.0
          DO 200 I=1,4
              TEMP1=TEMP1 + 0.5*(DB(3,I) + DE(3,I))*DEP(I)
  200     CONTINUE
          STOR(6)=TEMP1
      ELSE
C
C       Convert the 3-dimensional state to one of plane strain
C
          DO 300 I=1,4
              DB(3,I)=DB(4,I)
              DB(4,I)=0.0
              DE(3,I)=DE(4,I)
              DE(4,I)=0.0
  300     CONTINUE
          DO 400 I=1,3
              DB(I,3)=DB(I,4)
              DB(I,4)=0.0
              DE(I,3)=DE(I,4)
              DE(I,4)=0.0
  400     CONTINUE
      END IF
      RETURN
      END
```

```
      SUBROUTINE BOUND (PROP,VOID,X,XI,XJ,XIO,SIN3A,XKPB,BETA,
     *                  DFI,DFJ,DFAL,DFJJ)
C
C     Subroutine to evaluate relationship of current stress state
C                    to the bounding surface
C
      INTEGER IZONE
      REAL ALFUN,CV,RT,SINV
      REAL DFUN,FUN,FUNC
      REAL PROP(19),VOID,X,XI,XJ,XIO,SIN3A,XKPB,BETA,DFI,DFJ,DFJJ,DFAL,
     *     XN,DNAL,R,DRAL,A,DAAL,Y,C,ARB,BIG,SMALL,T,Q,QC,QO,FOP,XJO,
     *     BT,RHO,XIBAR,THETA,PSI,GAM,DYSAL,DFOPAL,DJOAL,DBTAL,DRHOAL,
     *     TEMP,TEMP1,TEMP2,TEMP3,TEMP4,TEMP5,TEMP6,TEMP7,TEMP8
C
      DATA ARB/0.001/, BIG/1.0E+20/, SMALL/1.0E-20/
      ALFUN(CV,RT,SINV)=2.0*RT*CV/(1.0 + RT - (1.0 - RT)*SINV)
      DFUN(FUN,RT,FUNC)=FUN**2*(1.0 - RT)/(2.0*RT*FUNC)
C
      XN=ALFUN(PROP(3),PROP(4),SIN3A)
      DNAL=DFUN(XN,PROP(4),PROP(3))
      R=ALFUN(PROP(9),PROP(12),SIN3A)
      DRAL=DFUN(R,PROP(12),PROP(9))
      A=ALFUN(PROP(10),PROP(13),SIN3A)
      DAAL=DFUN(A,PROP(13),PROP(10))
      Y=R*A/XN
      C=PROP(14)
C
C           Shift projection point
C
      TEMP1=XI - XIO*C
      IF(ABS(TEMP1) .LT. ARB) TEMP1=ARB
      TEMP2=C - 1.0/R
      TEMP3=TEMP1*TEMP2
      TEMP4=C*(C - 2.0/R)
      Q =XJ/TEMP1
      QC=XN/(1.0 - R*C)
      QO=BIG
      IF(C .NE. 0.0)  QO=XN*(SQRT(1.0 + Y*Y) - (1.0 + Y))/R/C
      IF(XJ .EQ. 0.0)  THEN
          IF(TEMP1 .GT. 0.0)  THEN
              IZONE=1
          ELSE
              IZONE=3
          END IF
      ELSE IF(C .GE. 1.0/R)  THEN
          IF(Q .GE. 0.0 .OR. Q .LE. QC)  THEN
              IZONE=1
          ELSE IF(Q .GE. QO)  THEN
              IZONE=3
          ELSE
              IZONE=2
          END IF
      ELSE
          IF(Q .GE. QC)  THEN
              IZONE=2
          ELSE IF(Q .GE. 0.0)  THEN
              IZONE=1
          ELSE IF(Q .LE. QO)   THEN
              IZONE=2
          ELSE
```

```
            IZONE=3
          END IF
        END IF
        IF(IZONE .EQ. 1)  THEN
C
C           Projection on ellipse 1
C
        TEMP5=TEMP1*TEMP1 + ((R - 1.0)*XJ/XN)**2
        BETA=XIO*(-TEMP3+SQRT(TEMP3*TEMP3-TEMP5*(TEMP4+(2.0-R)/R)))
     *       /TEMP5
        ELSE IF(IZONE .EQ. 2)  THEN
C
C           Projection on hyperbola
C
            TEMP5=TEMP4 - 2.0*A/R/XN
            TEMP6=XJ*(1.0/R + A/XN)/XN
            TEMP7=TEMP3 + TEMP6
            TEMP8=TEMP1*TEMP1 - (XJ/XN)*(XJ/XN)
            BETA=-0.5*XIO*TEMP5/TEMP7
            IF(TEMP8 .NE. 0.0)
     *           BETA=XIO*(-TEMP7 + SQRT(TEMP7*TEMP7 - TEMP8*TEMP5))/TEMP8
        ELSE
C
C           Projecton on ellipse 2
C
            T=PROP(11)
            TEMP5=SQRT(1.0 + Y*Y)
            FOP=XN/TEMP5
            XJO=A*(1.0 + Y - TEMP5)/Y
            BT=T*(XJO - T*FOP)/(XJO - 2.0*T*FOP)
            RHO=(BT - T)/FOP/XJO
            TEMP6=T - BT + C
            TEMP7=TEMP1*TEMP6
            TEMP8=TEMP1*TEMP1 + RHO*XJ*XJ
            BETA=XIO*(-TEMP7+SQRT(TEMP7*TEMP7
     *           -TEMP8*(TEMP6*TEMP6 - BT*BT/TEMP8
        END IF
C
C       Compute derivatives of the bounding surface w.r.t. invariants
C         and the value of the "bounding" plastic modulus for the
C                        appropriate zone.
C
        XIBAR=BETA*(XI - XIO*C) + XJ*C
        IF(XIBAR .EQ. 0.0)  XIBAR=SMALL
        THETA=BETA*XJ/XIBAR
        X=THETA/XN
        TEMP5=XIO
        IF(XIO .LT. PROP(7))  TEMP5=PROP(7)
        TEMP=12.0*VOID/(PROP(1) - PROP(2))**2/XIO*XIO*TEMP5
C
        IF(IZONE .EQ. 1)  THEN
C
C           Normal consolidation zone (ellipse 1)
C
            PSI=Y/((R - 1.0)*(R - 1.0))
            TEMP5=R*(1.0 + X*X + R*(R - 2.0)*X*X)
            GAM=(1.0 + (R - 1.0)*SQRT(1.0 + R*(R - 2.0)*X*X))/TEMP5
            DFI=2.0*XJ*(GAM - 1.0)*PSI
            DFJJ=2.0*XJO*GAM*(R - 1.0)/XN/XN*PSI*BETA/XIBAR
            DFJ=DFJJ*XJ
```

```
            XKPB=TEMP*(GAM - 1.0/R)*(GAM + R - 2.0)*PSI*PSI/R
            DFAL=PSI*6.0*(R-1.0)*THETA*GAM*XIO*(((R-1.0)/(R*R*
     *          (2.0/R-GAM-1.0)) + 1.0)*DRAL - (R-1.0)*DNAL/XN)/(XN*XN)
            RETURN
          ELSE IF(IZONE .EQ. 2)  THEN
C
C           Overconsolidated zone (hyperbola)
C
            TEMP5=1.0 - X*(1.0 + Y)
            GAM=-(TEMP5+SQRT((X-Y-1.0)**2 + (X*X-1.0)*Y*Y))/(R*(X*X-1.0))
            DFI=2.0*XIO*(GAM - 1.0/R)
            DFJ=2.0*XIO*((1.0 + Y)/R - X*GAM)/XN
            DFJJ=DFJ/XJ
            XKPB=TEMP*(GAM - 1.0/R)*(TEMP5*GAM + 2.0*A/XN)/R
            DFAL=6.0*XIO*(DNAL*(THETA*GAM/XN-1.0/R+A/(R*THETA*GAM)
     *         -2.0*A/XN)/(XN*XN)+DRAL*(1.0/THETA-1.0/XN+A/(XN*THETA*GAM))
     *         /(R*R) + DAAL*(1.0/XN - 1.0/(THETA*GAM*R))/XN)
            RETURN
          ELSE
C
C           Tension zone (ellipse 2)
C
            PSI=1.0/(R*(BT - T))
            GAM=(-T+BT-SQRT(BT*BT-RHO*THETA*THETA*T*(T-2.0*BT)))
     *           /(1.0 + RHO*THETA*THETA)
            DFI=2.0*PSI*XIO*(GAM + T - BT)
            DFJJ=2.0*PSI*XIO*GAM*RHO*BETA/XIBAR
            DFJ=DFJJ*XJ
            XKPB=TEMP*PSI*PSI*(GAM+T-BT)*(GAM*(BT-T) + T*(2.0*BT-T))
            DYSAL=Y*(DRAL/R + DAAL/A - DNAL/XN)
            DFOPAL=FOP*(DNAL/XN - Y*DYSAL/(1.0 + Y*Y))
            DJOAL=XJO*(DAAL/A - DYSAL/Y) + A*(1.0/Y - FOP/XN)*DYSAL
            DBTAL=((T-BT)*DJOAL - (T-2.0*BT)*T*DFOPAL)/(XJO-2.0*T*FOP)
            DRHOAL=DBTAL/FOP/XJO - RHO*(DFOPAL/FOP + DJOAL/XJO)
            DFAL=3.0*PSI*XIO*THETA*GAM*(DRHOAL + 2.0*RHO*DBTAL
     *           /(T+GAM-2.0*BT))
            RETURN
          END IF
          END
```

```
      SUBROUTINE JPLOT (NP,PINC,IX,IY,XT,YT)
C
C       Printer plotting subroutine written by J. S. De Natale.
C                 Modified by V. Kaliakin, 1983.
C
      CHARACTER*1 XT(8),YT(8),BUF(128)
      INTEGER ICPX(150),ICPY(150),ICEX(20),ICEY(20),I,J,IX,IY,IK,
     *        JK,NP,NEXP
      REAL PINC(150,8),XEXP(20),YEXP(20),XLAB(11),YLAB(11),XMIN,XMAX,
     *     YMIN,YMAX,XDIST,YDIST,XINC,YINC,TEMP
C
C         Read values for the experimental data points
C
      READ(5,902) NEXP
      IF(NEXP .NE. 0)  THEN
         READ(5,904) (XEXP(I),I=1,NEXP)
         READ(5,904) (YEXP(I),I=1,NEXP)
      END IF
C
C         Establish minimum and maximum axes values
C
      XMIN=PINC(1,IX)
      XMAX=PINC(1,IX)
      YMIN=PINC(1,IY)
      YMAX=PINC(1,IY)
      DO 100 I=2,NP
          IF(XMIN .GT. PINC(I,IX))  XMIN=PINC(I,IX)
          IF(XMAX .LT. PINC(I,IX))  XMAX=PINC(I,IX)
          IF(YMIN .GT. PINC(I,IY))  YMIN=PINC(I,IY)
          IF(YMAX .LT. PINC(I,IY))  YMAX=PINC(I,IY)
  100 CONTINUE
      IF(NEXP .NE. 0)  THEN
          DO 200 I=1,NEXP
              IF(XMIN .GT. XEXP(I))  XMIN=XEXP(I)
              IF(XMAX .LT. XEXP(I))  XMAX=XEXP(I)
              IF(YMIN .GT. YEXP(I))  YMIN=YEXP(I)
              IF(YMAX .LT. YEXP(I))  YMAX=YEXP(I)
  200     CONTINUE
      END IF
C
C        Establish adjusted maximum and minimum axes values
C
      CALL SHIFT (XMIN,XMAX)
      CALL SHIFT (YMIN,YMAX)
C
C        Establish axes labels
C
      XDIST=XMAX-XMIN
      YDIST=YMAX-YMIN
      XINC=XDIST/10.0
      YINC=YDIST/10.0
      XLAB( 1)=XMIN
      XLAB(11)=XMAX
      YLAB( 1)=YMIN
      YLAB(11)=YMAX
C
      DO 300 I=2,10
          TEMP=FLOAT(I-1)
          XLAB(I)=XMIN + XINC*TEMP
          YLAB(I)=YMIN + YINC*TEMP
```

```
  300 CONTINUE
C
C        Establish data point coordinates
C
      DO 400 I=1,NP                                  ! For computed data points
          TEMP=(PINC(I,IX) - XMIN)/XDIST*100.0
          CALL SETCRD (I,TEMP,ICPX)
          TEMP=ABS(PINC(I,IY) - YMAX)/YDIST* 50.0
          CALL SETCRD (I,TEMP,ICPY)
  400 CONTINUE
      IF(NEXP .NE. 0)  THEN                          ! For experimental data points
          DO 500 I=1,NEXP
              TEMP=(XEXP(I) - XMIN)/XDIST*100.0
              CALL SETCRD (I,TEMP,ICEX)
              TEMP=ABS(YEXP(I) - YMAX)/YDIST*50.0
              CALL SETCRD (I,TEMP,ICEY)
  500     CONTINUE
       END IF
C
C           Printer plot the data points
C
      JK=0
      WRITE(6,900)
      DO 800 IK=0,50              ! Loop through each of the print lines
          CALL BZERO (BUF,128)
C
          DO 600 J=1,NP                      ! Print computed data points
              IF(ICPY(J) .EQ. IK)  THEN
                  BUF(ICPX(J) + 25)='*'
              END IF
  600     CONTINUE
          IF(NEXP .NE. 0)  THEN              ! Print experimental data points
              DO 700 J=1,NEXP
                  IF(ICEY(J) .EQ. IK)  THEN
                      BUF(ICEX(J) + 25)='#'
                  END IF
  700         CONTINUE
          END IF
          CALL BORDER (JK,IK,XLAB,YLAB,XT,YT,BUF)
  800 CONTINUE
C
  900 FORMAT(1H1,/)
  902 FORMAT(I5)
  904 FORMAT(8E10.3)
      RETURN
      END
```

```
      SUBROUTINE SHIFT (PMIN,PMAX)
C
C       Subroutine to compute adjusted maximum amd minimum axes values.
C                       Written by J. S. De Natale.
C
      INTEGER II,JJ,NN
      REAL PMIN,PMAX,DX,XL,XU,D0,D1,C1,DU,DL
C
      DX=(PMAX-PMIN)/10.0
      XL=PMIN-DX
      XU=PMAX+DX
      D0=ABS(PMIN)
      D1=ABS(PMAX)
      IF(D1 .LT. D0)  D1=D0
      NN=21
      DO 100 JJ=1,40
          NN=NN-1
          C1=10.0**NN
          IF(D1 .GE. C1)  GOTO 200
  100 CONTINUE
  200 CONTINUE
      D1=PMAX/C1
      II=IFIX(D1)
      DU=FLOAT(II)*C1
      IF(DU .LT. PMAX)  II=II+1
      DU=FLOAT(II)*C1
      IF(DU .GT. XU)  THEN
          C1=C1/10.0
          GOTO 200
      END IF
C
      D1=PMIN/C1
      II=IFIX(D1)
      DL=FLOAT(II)*C1
      IF(DL .GT. PMIN)  II=II-1
      DL=FLOAT(II)*C1
      IF(DL .LT. XL)  THEN
          C1=C1/10.0
          GOTO 200
      END IF
      PMIN=DL
      PMAX=DU
C
      RETURN
      END
```

```
      SUBROUTINE BORDER (JK,IK,XLAB,YLAB,XT,YT,BUF)
C
C         Subroutine to provide borders for printer plotting
C
      CHARACTER*1 XT(8),YT(8),BUF(128),TEMP(10)
      INTEGER I,J,JK,IK,ITEMP,NUMLAB
      REAL XLAB(11),YLAB(11)
C
      PARAMETER (no=0)
      PARAMETER (yes=1)
      NUMLAB=no
C
      IF(IK .EQ. 0 .OR. MOD(IK,5) .EQ. 0)  THEN
          NUMLAB=yes                          ! Numeric label required
          JK=JK + 1
      END IF
C
      IF(IK .EQ. 0 .OR. IK .EQ. 50)  THEN     ! Top or bottom border
          ENCODE (10,900,TEMP)  YLAB(12-JK)
          DO 100 I=1,10
              BUF(I+12)=TEMP(I)
  100     CONTINUE
          DO 300 I=2,11
              DO 200 J=6,14
                  ITEMP=I*10 + J
                  IF(BUF(ITEMP) .EQ. ' ')  BUF(ITEMP)='-'
  200         CONTINUE
              ITEMP=I*10 + 5
              IF(BUF(ITEMP) .EQ. ' ')  BUF(ITEMP)='I'
  300     CONTINUE
          NUMLAB=no
      ELSE
          IF(IK .EQ. 25)  THEN                ! Title for vertical axis
              DO 400 I=1,8
                  BUF(I+2)=YT(I)
  400         CONTINUE
          END IF
      END IF
C
      IF(NUMLAB .EQ. yes)  THEN               !   Numeric labels and
          ENCODE (10,900,TEMP)  YLAB(12-JK)   ! associated symbols, but
          DO 500 I=1,10                       ! not for top & bot. borders
              BUF(I+12)=TEMP(I)
  500     CONTINUE
          IF(BUF(25)  .EQ. ' ')  BUF(25) ='+'
          IF(BUF(26)  .EQ. ' ')  BUF(26) ='-'
          IF(BUF(124) .EQ. ' ')  BUF(124)='-'
          IF(BUF(125) .EQ. ' ')  BUF(125)='+'
      ELSE
          IF(BUF(25)  .EQ. ' ')  BUF(25) ='I'
          IF(BUF(125) .EQ. ' ')  BUF(125)='I'
      END IF
      WRITE(6,904) (BUF(I),I=1,125)           ! Print contents of buffer
C
      IF(IK .EQ. 50)  THEN
          CALL BZERO (BUF,128)                ! Get labels for bottom border
          DO 600 I=1,11
              ENCODE (9,902,TEMP)  XLAB(I)
              DO 600 J=1,9
                  ITEMP=I*10 + 9 + J
```

```
                BUF(ITEMP)=TEMP(J)
  600       CONTINUE
            WRITE(6,904) (BUF(I),I=1,128)
C
            WRITE(6,906)                          ! Print a blank line
C
            CALL BZERO (BUF,128)
            DO 700 I=1,8                          ! Title for bottom border
                BUF(I+71)=XT(I)
  700       CONTINUE
            WRITE(6,904) (BUF(I),I=1,79)
        END IF
C
  900 FORMAT(F10.4)
  902 FORMAT (F9.4)
  904 FORMAT(128A1)
  906 FORMAT(/)
        RETURN
        END
```

```
      SUBROUTINE SETCRD (IK,TEMP1,ICXY)
C
C        Subroutine to compute rounded (integer) data point coordinates
C
      INTEGER IK,ICXY(1)
      REAL TEMP1,TEMP2
C
      IF(TEMP1 .LT. 0.0)  TEMP1=0.0
      TEMP2=AINT(TEMP1)
      IF(TEMP1 - TEMP2 .GT. 0.5)  TEMP2=TEMP2 + 1.0
      ICXY(IK)=IFIX(TEMP2)
C
      RETURN
      END
```

```
      SUBROUTINE BZERO (B,N)
C
C         Subroutine to initialize a character array
C
      CHARACTER*1 B(N)
      INTEGER I,N
C
      DO 100 I=1,N
          B(I)=' '
  100 CONTINUE
      RETURN
      END
```

Appendix IV: Listing of Program EVAL

```
C                          ***** EVAL *****
C
C     Program to predict homogeneous test results for material models.
C        Prepared by L.R. Herrmann at the University of California,
C              Davis campus. Fortran 77 version, revised 1983.
C
      INTEGER I,J,K,NJ,K7,ITNO,NUM,INC,NINC,IPRNT1,IPRNT2,ITMAX,KIND,
     *     LARGE,LOCIT,JUNK,IFLAG,IPRINT,ITITLE(40),ICOD(7),IPLOT(10)
      REAL PROP(19),STOR(7),SIGB(6),DSIG(6),EPB(6),DEP(6),V(6),DV(6),
     *     RP(6),R(7),DB(6,6),DE(6,6),S(7,7),PINC(150,8),CONFIN,
     *     ERMAX,CONFL,SNORM1,SNORM2,ENORM1,ENORM2,DLTAU,U,D,GAM,
     *     TEMP,TEMP1,TEMP1T
      PARAMETER (yes=1)
      PARAMETER (no=0)
C
    1 CONTINUE
      READ(5,800,END=999) (ITITLE(I),I=1,40)
      WRITE(6,900) (ITITLE(I),I=1,40)
      READ(5,802)  STOR(1),STOR(7),CONFIN
C
C          Read material properties
C
      CALL RPROP (PROP)
C
C          Read iteraton and analysis information
C
      READ(5,801) ITMAX,ERMAX,CONFL,KIND,LARGE,LOCIT
      IF(ITMAX .GT. 20)  ITMAX=20
      IF(CONFL .LE. 0.0) CONFL=0.3
      IF(LOCIT .EQ. 0)   LOCIT=1
C
      READ(5,804)  IPRNT1,IPRNT2        ! Read flags for output control
      READ(5,804) (IPLOT(I),I=1,10)     ! Read plotting instructions
C
      WRITE(6,902) STOR(7),CONFIN,STOR(1)
      STOR(1)=STOR(1)*3.0
      IF(KIND  .EQ. 0)   WRITE(6,906)
      IF(KIND  .EQ. 1)   WRITE(6,907)
      IF(LARGE .EQ. 0)   WRITE(6,908)
      IF(LARGE .EQ. 1)   WRITE(6,909)
      WRITE(6,901) ITMAX,LOCIT,ERMAX,CONFL
C
C           Initialization
C
      NUM=0
      IFLAG=no
      IPRINT=no
      ICOD(7)=0                 ! Fictitious increment type specification
      TEMP1T=0.0
      SNORM1=0.0
      ENORM1=0.0
      DLTAU=0.0
      U=0.0
      DO 100 I=1,6
          SIGB(I)=0.0
          DSIG(I)=0.0
          EPB(I)= 0.0
          DEP(I)= 0.0
  100 CONTINUE
```

```
      DO 150 I=1,3
          SIGB(I)=CONFIN
  150 CONTINUE
C
C        Loading segment loop
C
      IF(IPRNT1 .EQ. 0)  WRITE(6,903)
  200 CONTINUE
      READ(5,803) (ICOD(I),V(I),I=1,6),NINC,D
      IF(ICOD(1) .GT. 1)  GOTO 750
      IF(D .EQ. 0.0)  D=1.0
C
C        Determine first increments
C
      TEMP1=1.0/(FLOAT(NINC))
      IF(D .NE. 1.0)  TEMP1=(1.0 - D)/(1.0 - D**NINC)
      DO 250 I=1,6
          TEMP=V(I) - SIGB(I)
          IF(ICOD(I) .EQ. 1)  TEMP=V(I) - EPB(I)
          DV(I)=TEMP*TEMP1
  250 CONTINUE
C
C     Change sign of strain estimate at beginning of new segment
C    in case of unstable behavior at the end of the previous one.
C
      DLTAU=DLTAU*0.01
      DO 300 I=1,6
          DSIG(I)=DSIG(I)*.01
          DEP(I)=DEP(I)*-.01
  300 CONTINUE
C
C          Increment loop
C
      DO 700 INC=1,NINC
C
C          Iteration loop (successive approximation)
C
          DO 600 ITNO=1,ITMAX
C
C          Get incremental properties
C
              NJ=NUM + 1
              K7=ITNO
              CALL CLAY (3,NJ,K7,ERMAX,PROP,STOR,SIGB,EPB,DSIG,DEP,
     *                   DB,DE,U,DLTAU,GAM,KIND,LARGE,LOCIT,0.5)
C
C         Form and modify stiffness
C
              DO 350 I=1,3
                  S(7,I)=GAM
                  S(7,I+3)=0.0
                  S(I,7)=1.0
                  S(I+3,7)=0.0
  350         CONTINUE
              S(7,7)=-1.0
              R(7)=0.0
              DO 450 I=1,6
                  DO 400 J=1,6
                      S(I,J)=0.5*(DB(I,J) + DE(I,J))
  400             CONTINUE
```

```
                R(I)=DV(I)
                IF(ICOD(I) .EQ. 1)  THEN
                    DO 420 K=1,7
                        IF(ICOD(K) .EQ. 0)
     *                      R(K)=R(K) - S(K,I)*DV(I)
                        S(I,K)=0.0
                        S(K,I)=0.0
  420                   CONTINUE
                        S(I,I)=1.0
                    END IF
  450           CONTINUE
C
C           Solve for strain increment
C
              CALL SOLVE (KIND,ICOD,S,R)
C
C         Calculate total stress increments
C
              DO 550 I=1,6
                  TEMP=0.0
                  IF(I .LT. 4)  TEMP=R(7)
                  DO 500 J=1,6
                      TEMP=0.5*(DB(I,J) + DE(I,J))*R(J) + TEMP
  500             CONTINUE
                  RP(I)=TEMP
  550           CONTINUE
C
C                 Compute error norms
C
              CALL ERNORM (IFLAG,IPRINT,ITNO,NUM,KIND,ERMAX,CONFL,
     *                     SNORM1,SNORM2,ENORM1,ENORM2,TEMP1T,RP,
     *                     DSIG,R,DEP,DLTAU)
              IF(IFLAG .EQ. yes)  GOTO 640
  600       CONTINUE
            IPRINT=yes
            CALL ERNORM (IFLAG,IPRINT,ITNO,NUM,KIND,ERMAX,CONFL,SNORM1,
     *                   SNORM2,ENORM1,ENORM2,TEMP1T,RP,DSIG,R,DEP,DLTAU)
C
C             Upon failure to converge, eat up remaining input data
C
  620       READ(5,803) JUNK
            IF(JUNK .GT. 1)  GOTO 750
            GOTO 620
C
C             Update total values
C
  640       NUM=NUM + 1
            TEMP1T=0.0
            DO 660 I=1,6
                DV(I)=DV(I)*D
                DSIG(I)=RP(I)
                DEP(I) =R(I)
                SIGB(I)=SIGB(I) + DSIG(I)
                EPB(I) =EPB(I)  + DEP(I)
                TEMP1T =TEMP1T  + ABS(SIGB(I))*0.1
  660       CONTINUE
            IF(KIND .EQ. 0)  DLTAU=R(7)
            U=U + DLTAU
C
C         Store incremental values for future plotting
```

```
C
          PINC(NUM, 1)=(SIGB(3) - SIGB(1))
          PINC(NUM, 2)=(SIGB(1) + SIGB(2) + SIGB(3))/3.0 - U
          PINC(NUM, 3)= U
          PINC(NUM, 4)=(EPB(1) + EPB(2) + EPB(3))*100.0
          PINC(NUM, 5)=(EPB(3) - EPB(1))/(1.50)*100.0
          PINC(NUM, 6)= EPB(3)*100.0
          PINC(NUM, 7)=(SIGB(3) - SIGB(1))/(STOR(1)/3.0)
          PINC(NUM, 8)=(SIGB(1) + SIGB(2) + SIGB(3) - 3.0*U)/STOR(1)

C
C        Print incremental values of stresses and strains if desired
C
          IF(IPRNT1 .EQ. 0)
     *            WRITE(6,904) NUM,(EPB(I),I=1,6),(SIGB(I),I=1,6),U,ITNO
  700 CONTINUE
      WRITE(6,905)
      GOTO 200
  750 CONTINUE
C
C            Print computed incremental parameters if desired
C
      IF(IPRNT2 .EQ. 0) THEN
          WRITE(6,910)
          DO 780 I=1,NUM
              WRITE(6,911) I,PINC(I,6),(PINC(I,J),J=1,5)
  780     CONTINUE
      END IF
      IF(IPLOT( 1) .EQ. yes)
     *          CALL JPLOT(NUM,PINC,2,1,'    P   ','    Q   ')
      IF(IPLOT( 2) .EQ. yes)
     *          CALL JPLOT(NUM,PINC,6,1,'  EPS-1 ','    Q   ')
      IF(IPLOT( 3) .EQ. yes)
     *          CALL JPLOT(NUM,PINC,6,3,'  EPS-1 ','    U   ')
      IF(IPLOT( 4) .EQ. yes)
     *          CALL JPLOT(NUM,PINC,6,4,'  EPS-1 ','  E-VOL ')
      IF(IPLOT( 5) .EQ. yes)
     *          CALL JPLOT(NUM,PINC,8,7,'   P/PO ','   Q/PO ')
      IF(IPLOT( 6) .EQ. yes)
     *          CALL JPLOT(NUM,PINC,6,7,'  EPS-1 ','   Q/PO ')
      GOTO 1
  999 CALL EXIT
C
C               Format statements
C
  800 FORMAT(40A2)
  801 FORMAT(I5,2E10.3,3I5)
  802 FORMAT(3E10.3)
  803 FORMAT(6(I1,E9.1),I5,E10.3)
  804 FORMAT(10I5)
  900 FORMAT(1H1,/,4X,40A2,4(/))
  901 FORMAT(/,10X,'ITERATION AND CONVERGENCE PARAMETERS:',/,
     1        24X,'ITMAX=',I3,/,24X,'LOCIT=',I3,/,
     2        24X,'ERMAX=',F6.3,/,24X,'CONFL=',F6.3,/)
  902 FORMAT(/,29X,'INITIAL VOID RATIO =',F7.4,/,
     1       21X,'INITIAL CONFINING PRESSURE =',1PE12.3,/,
     2       10X,'INITIAL PRECONSOLIDATION PRESSURE, PO =',1PE12.3,//)
  903 FORMAT(1H1,3X,'N',4X,'EPS-X',4X,'EPS-Y',4X,'EPS-Z',3X,'GAM-XY',
     1       3X,'GAM-XZ',3X,'GAM-YZ',5X,'SIG-X',5X,'SIG-Y',5X,'SIG-Z',
     2       4X,'TAU-XY',4X,'TAU-XZ',4X,'TAU-YZ',6X,'U',3X,'IT #',/)
```

```
904 FORMAT(1X,I3,1P6E9.1,7E10.2,1X,I3)
905 FORMAT(/)
906 FORMAT(5X,'*****  REFORMULATED NEARLY-INCOMPRESSIBLE ',
   1       'ANALYSIS  *****'/)
907 FORMAT(5X,'*****  NON-REFORMULATED NEARLY-INCOMPRESSIBLE ',
   1       'ANALYSIS  *****'/)
908 FORMAT(5X,'*****  ENGINEERING STRESSES AND STRAINS ',
   1       'ASSUMED  *****'/)
909 FORMAT(5X,'*****  TRUE STRESSES AND NATURAL STRAINS ',
   1       'ASSUMED  *****'/)
910 FORMAT(1H1,//,9X,'N',5X,'EPS-1',9X,'Q',9X,'P',9X,'U',
   1        5X,'E-VOL',5X,'E-DEV'/9X,'-',5X,'-----',5X,'-----',
   2        5X,'-----',5X,'-----',5X,'-----',5X,'-----'/)
911 FORMAT(5X,I5,6F10.2)
    END
```

```
      SUBROUTINE SOLVE (KIND,ICOD,S,R)
C
C        Subroutine to solve the 7 x 7 stiffness matrix for EVAL
C
      INTEGER I,II,IL,J,K,N,KIND,ICOD(7)
      REAL TEMP,R(7),S(7,7)
C
C          Reduction of stiffness matrix and r.h.s. vector
C
      N=7-KIND
      DO 400 I=1,N
          IF(ICOD(I) .EQ. 0) THEN          ! Avoid work for trivial rows
              TEMP=1.0/S(I,I)
              R(I)=R(I)*TEMP
              DO 100 J=I,N
                  S(I,J)=S(I,J)*TEMP
  100         CONTINUE
              IF(I .NE. N)  THEN
                  II=I+1
                  DO 300 J=II,N
                      IF(ICOD(I) .EQ. 0)  THEN
                          TEMP=-S(J,I)
                          DO 200 K=I,N
                              S(J,K)=S(J,K) + TEMP*S(I,K)
  200                     CONTINUE
                          R(J)=R(J) + TEMP*R(I)
                      END IF
  300             CONTINUE
              END IF
          END IF
  400 CONTINUE
C
C              Back substitution
C
      I=N
      DO 500 II=2,N
          I=I-1
          IL=I+1
          DO 500 J=IL,N
              R(I)=R(I) - S(I,J)*R(J)
  500 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE ERNORM (IFLAG,IPRINT,ITNO,NUM,KIND,ERMAX,CONFL,SNORM1,
     *                   SNORM2,ENORM1,ENORM2,TEMP1T,RP,DSIG,R,DEP,DLTAU)
C
C         Subroutine to check for convergence of the incremental solution
C
      INTEGER I,IFLAG,IPRINT,ITNO,NUM,KIND
      REAL ERSIG,EREPS,ERMAX,RP(6),DSIG(6),R(7),DEP(6),DLTAU,TEMP1,
     *     TEMP2,TEMP1T,CONFS,CONFE,CONFL,SNORM1,SNORM2,ENORM1,ENORM2
      PARAMETER (yes=1)
      PARAMETER (no=0)
C
      ERSIG=0.0                    ! Compute error norms
      EREPS=0.0
      TEMP1=0.0
      TEMP2=0.0
      DO 100 I=1,6
          ERSIG=ERSIG + ABS(RP(I)-DSIG(I))
          EREPS=EREPS + ABS(R(I)-DEP(I))
          TEMP1=TEMP1 + ABS(RP(I))
          TEMP2=TEMP2 + ABS(R(I))
  100 CONTINUE
      IF(TEMP1 .LT. TEMP1T)  THEN
          ERSIG=ERSIG/TEMP1T
      ELSE
          ERSIG=ERSIG/TEMP1
      END IF
      EREPS=EREPS/TEMP2
C
C         Upon failure to converge, print pertinent information about errors
C
      IF(IPRINT .EQ. yes)  THEN
          WRITE(6,900) NUM+1,ERSIG,EREPS,ERMAX
          RETURN
      END IF
C
C       Check norms against error tolerance to determine convergence
C
      IF(ERSIG .GE. ERMAX .OR. EREPS .GE. ERMAX) THEN
C
          CONFS=1.0        ! Apply Aitken's convergence acceleration
          CONFE=1.0
          IF(ITNO .NE. 1  .AND.  (-1)**ITNO .LT. 0)  THEN
              CALL ACCEL (SNORM2,SNORM1,TEMP1,CONFS,CONFL)
              CALL ACCEL (ENORM2,ENORM1,TEMP2,CONFE,CONFL)
          END IF
          TEMP1=0.0
          TEMP2=0.0
          DO 200 I=1,6
              DSIG(I)=RP(I)*CONFS + (1.0-CONFS)*DSIG(I)
              DEP(I) =R(I)*CONFE  + (1.0-CONFE)*DEP(I)
              TEMP1=TEMP1 + ABS(DSIG(I))
              TEMP2=TEMP2 + ABS(DEP(I))
  200     CONTINUE
          IF(KIND .EQ. 0)  DLTAU=R(7)*CONFS + (1.0-CONFS)*DLTAU
          SNORM2=SNORM1
          ENORM2=ENORM1
          SNORM1=TEMP1
          ENORM1=TEMP2
          IFLAG=no
      ELSE
```

```
          IFLAG=yes
        END IF
C
  900 FORMAT(///,3X,'***** CONVERGENCE DID NOT OCCUR FOR INCREMENT ',I3,
     1       ' *****',/,10X,'ERSIG=',1PE10.3,3X,'EREPS=',1PE10.3,
     2                  5X,'ERMAX=',1PE10.3,///)
        RETURN
        END
```

```
      SUBROUTINE ACCEL (X2,X1,X,C,XL)
C
C     This subroutine calculates the Aitken's convergence factor
C
      REAL X,X1,X2,XL,C,TEMP
      C=1.0
      TEMP=-X2 + 2.0*X1 - X
      IF(TEMP .NE. 0.0)  THEN
          C=(X1 - X2)/TEMP
          IF(C .LT. XL)       C=XL
          IF(C .GT. 1.0/XL)   C=1.0/XL
      END IF
      RETURN
      END
```

```
      SUBROUTINE RPROP (PROP)
C
C     This subroutine reads in and modifies the parameters required
C      by the bounding surface plasticity model for cohesive soils.
C
      REAL PROP(19)
      READ(5,800) (PROP(I),I=1,3),(PROP(I),I=9,11),PROP(7),PROP(5),
     1             PROP(8),PROP(6), PROP(17), PROP(16), PROP(19),
     2             PROP(4),PROP(18),(PROP(I),I=12,15)
      WRITE(6,900) PROP(1), PROP(3), PROP(2), PROP(4)
      IF(PROP(5) .LT. 0.5)  WRITE(6,901) PROP(5),PROP(7)
      IF(PROP(5) .GE. 0.5)  WRITE(6,902) PROP(5),PROP(7)
      WRITE(6,903) PROP(17),PROP(19),PROP(16),PROP(18)
      WRITE(6,904) PROP(9), PROP(12),PROP(10),PROP(13),PROP(11)
C
C       Check the magnitude of the shape parameter "T"
C
      CALL TCHECK (PROP)
      WRITE(6,905) PROP(15),PROP(14),PROP(8)
      IF(PROP(6) .EQ. 0.0)  WRITE(6,906)
      IF(PROP(6) .NE. 0.0)  WRITE(6,907) PROP(6)
C
C        Convert parameters from triaxial to invariant stress space
C
      PROP(3) =PROP(3)/SQRT(27.0)
      PROP(7) =PROP(7)*3.0
      RETURN
C
C             Format statements for RPROP
C
  800 FORMAT(8E10.3)
  900 FORMAT(10X,'CLASSICAL CLAY MATERIAL CONSTANTS:',/,
     1       15X,'LAMBDA =',F7.4,17X,'MC =',F7.4,/,
     2       15X,'KAPPA  =',F7.4,14X,'ME/MC =',F7.4,/)
  901 FORMAT(15X,'POISSON*S RATIO =',F7.4,8X,'PL =',1PE10.3,//)
  902 FORMAT(15X,'SHEAR MODULUS =',1PE10.3,7X,'PL =',1PE10.3,//)
  903 FORMAT(10X,'HARDENING PARAMETERS:',/,
     1       15X,'SM =',F7.4,21X,'H2 =',F7.4,/,
     2       15X,'HC =',F7.4,18X,'HE/HC =',F7.4,//)
  904 FORMAT(10X,'PARAMETERS DESCRIBING SHAPE OF BOUNDING SURFACE:',/,
     1       15X,'RC =',F7.4,18X,'RE/RC =',F7.4,/,
     2       15X,'AC =',F7.4,18X,'AE/AC =',F7.4,/,
     3       16X,'T =',F7.4,//)
  905 FORMAT(17X,'ELASTIC NUCLEUS PARAMETER, S =',F7.4,/,
     1       15X,'PROJECTION CENTER PARAMETER, C =',F7.4,//,
     2       25X,'ATMOSPHERIC PRESSURE =',1PE10.3,/)
  906 FORMAT(//,25X,'***** DRAINED CONDITIONS *****',/)
  907 FORMAT(//,5X,'***** UNDRAINED CONDITIONS - THE COMBINED ',
     1       'SKELETON AND WATER BULK MODULUS =',1PE10.3,' *****',/)
      END
```

```
      SUBROUTINE TCHECK (PROP)
C
C        This subroutine checks the value of the bounding surface shape
C   parameter "T" and adjusts this value if it exceeds the theoretical max.
C              Original version written by J.S. De Natale.
C
      REAL TEMP1,TEMP2,TEMPR,PROP(19)
      YFUN(TT)=(1.0 + TT)*SQRT(1.0 + TT*TT) - (1.0 + TT*TT)
C
C        Check against theoretical limit in compression
C
      TEMP1=PROP(11)
      TEMP2=PROP(9)*PROP(10)*SQRT(27.0)/PROP(3)
      TEMP2=YFUN(TEMP2)
      TEMP2=TEMP2/2.0/PROP(9)
      IF(PROP(11) .GT. TEMP2)  PROP(11)=TEMP2
C
C        Check against theoretical limit in extension
C
      TEMPR=PROP(9)*PROP(12)
      TEMP2=TEMPR*PROP(10)*PROP(13)*SQRT(27.0)/PROP(3)/PROP(4)
      TEMP2=YFUN(TEMP2)
      TEMP2=TEMP2/2.0/TEMPR
      IF(PROP(11) .GT. TEMP2)  PROP(11)=TEMP2
      IF(PROP(11) .NE. TEMP1)  WRITE(6,100) PROP(11)
  100 FORMAT(5X,'*****  THE USER-SPECIFIED VALUE OF T EXCEEDS THE MAX',
     1     ' PERMISSIBLE VALUE  *****',/,5X,'*****        T HAS THUS',
     2     ' BEEN AUTOMATICALLY RESET TO  T =',F7.4,8X,' *****',//)
C
      RETURN
      END
```


DATE:
ILME